Extreme Programming

– användbarhet i praktiken

Kandidatuppsats, 15 högskolepoäng, INFK01 i informatik

Framlagd: juni, 2009

Författare: Björn Glöck
Gustav Nilsson

Handledare: Claus Persson

Examinatorer: Agneta Olerup
Erik Wallin

Abstract

Titel	Extreme Programming – användbarhet i praktiken
Författare	Björn Glöck Gustav Nilsson
Utgivare	institutionen för informatik
Handledare	Claus Persson
Examinator	Agneta Olerup Erik Wallin
Publiceringsår	2009
Uppsatstyp	kandidatuppsats
Språk	svenska
Nyckelord	extreme programming**,** xp, agil, användbarhet, systemutvecklare, organisation

Abstract

Extreme Programming rönte stora framgångar i början av 2000-talet. På senare tid har metoden blivit allt mindre populär. Flera författare har pekat på problematiken med att använda XP. Syftet med studien var att undersöka den upplevda användbarheten kring XP, utifrån utvecklaren och organisationens perspektiv. Vi har konstruerat ett teoretiskt ramverk baserat på användbarhet, systemutvecklingsmetoder i praktiken samt utvecklingens kontext. Vårt tillvägagångssätt har varit kvalitativt där vi har utfört fyra semistrukturerade intervjuer på individer som har arbetat med XP på olika sätt. Resultatet av empirin relaterades därefter mot vårt teoretiska ramverk. Vi fann brister i XP-metodens upplevda användbarhet. Ur ett organisatoriskt perspektiv visar metoden på bristande projektstyrningsstöd samt att vissa praktiker var svåra för organisationen att acceptera. Metoden har visat sig ställa stora krav på utvecklarna genom att kräva en stor kunskapsbredd samt att vissa delar av metoden var svåra att leva upp till i praktiken.

Innehåll

1 Inledning

1.1 Bakgrund

Systemutvecklingsmetoder har förekommit sedan 1960-talet och har haft som uppgift att lösa de problem som fanns inom mjukvaruindustrin i form av försenad leverans, undermålig kvalitet och överskriden budget. Dessa problem existerar än idag, trots att åtskilliga systemutvecklingsmetoder har lanserats under årens lopp. (Fitzgerald, Hartnett, & Conboy, 2006) Extreme Programming (XP) lanserades år 1999 av Kent Beck. Metoden anammade det populära *agila* synsättet, där nyckelorden var *flexibilitet* och *anpassning*. Syftet var att utvecklingsprojekten skulle kunna hantera fortlöpande kravförändringar. Bakgrunden var att kraven på systemen ofta ändrades under arbetets gång på ett oförutsägbart sätt och därmed efterfrågandes systemutvecklingsmetoder som kunde hantera detta, då existerande metoder inte kunde hantera detta på ett tillfredställande sätt. Beck lanserade sin bakomliggande filosofi som inbegrep de värderingar och principer som systemutvecklingsarbetet enligt honom behövde följa för att lyckas med detta. Värderingarna och principerna följdes upp av en samling konkreta praktiker och roller som skulle tillämpas inom utvecklingsgruppen. (Beck, 1999; Beck, 2000) XP fick snabbt stor uppmärksamhet och flera rapporter visade i början på 2000-talet på lyckade försök att tillämpa metoden. Framgången ledde till att den växte till att bli den allra mest kända och populära agila metoden. (Abrahamsson, 2003)

Ungefär tio år senare är metodens popularitet och framgång inte lika framträdande. Tvärtom förekommer det under de senaste åren flera rapporter som pekar på XP:s otillräcklighet vid praktisk tillämpning. Kritik riktas mot det holistiska synsättet som kan te sig alldeles för idealistiskt. Beck hävdar nämligen att samtliga delar av XP måste tillämpas fullt ut för att metoden ska ge bästa resultat. (Beck, 2000; Mirakhorli et al., 2008; Kennan, 2004) Fitzgerald et al. (2006) talar istället om hur metoden behöver *anpassas* till den unika systemutvecklingsmiljön och dessutom behöver den kompletteras med fler projektstyrningsfunktioner för att passa in i organisationen. Detta resonemang om att anpassa metoden får stöd av Luck (2004) som dock understryker att det samspel som finns mellan XP:s olika praktiker innebär att en anpassning måste ske på ett väl genomtänkt sätt för att fungera. Skulle någon praktik inte tillämpas behöver bortfallet kompenseras för att metoden inte skall tappa sin synergieffekt. (Hussain et al., 2008; Mirakhorli et al., 2008) När det istället handlar om en komplett tillämpning av XP finns det även här dokumenterat en del svårigheter.

I de fall där utvecklare som aldrig tidigare har använt metoden tillämpar den, finns det flertalet bevis på att detta har lyckats väl, men även här har en fullskalig tillämpning visat sig vara problematisk. (Hedin et al., 2003; Lawrence, 2004; Karlström, 2001) Trots att exempelvis Lawrence (2004) hävdar att det största misstaget som kan göras när juniora utvecklare skall tillämpa metoden, är att inte tillämpa metoden fullt ut, så ställer den fullskaliga tillämpningen ändå problem. Det anses svårt att få alla delar att fungera som det är tänkt och vidare anses vissa delar vara viktigare än andra.

XP grundar sig på fyra värderingar, vilka transformeras till femton principer för att slutligen skapa tolv praktiker. Vidare finns ett antal rollanvisningar som ska följas och det omfattande antalet delkomponenter syftar till att möta de situationer som utvecklingsgruppen behöver

hantera och bemästra i en snabbt föränderlig systemutvecklingsmiljö. Syftet för Beck har varit att skapa en komplett och heltäckande systemutvecklingsmetod. (Beck, 2000) Denna mängd krav på systemutvecklarna väcker intresse angående den påverkan de har, för den av utvecklarna upplevda användbarheten med metoden. Genast riktas fokus mot den problematik som ligger bakom att metoden är svår att implementera i sin helhet och en utredning av den kan leda till ökad kunskap om i vilka situationer metoden är tillämpbar, samt vilka förutsättningar som underlättar användandet för att öka sannolikheten för ett fördelaktigt resultat. Denna typ av information kan underlätta för näringslivet vid val av systemutvecklingsmetod och även för institutionen för informatik ligger det i intresse att få en djupare förståelse av systemutvecklingsmetoder i praktiken.

1.2 Problemformulering

Det faktum att utvecklingen av metoden har gått mot att skräddarsy den vid tillämpning, samtidigt som det finns flera fall där den på ett framgångsrikt sätt implementerats i sin helhet, väcker funderingar. En ökad insikt i de erfarenheter som skapades i dessa projekt, kan generera förklaringar till metodens tillämpbarhet och i förlängningen vilka förutsättningar som krävs för att tillämpa den. Mängden praktiker som skall uppfyllas gör det problematiskt med att infria alla. (Karlström, 2002; Abrahamsson & Koskela, 2004). Detta är faktorer som mycket väl kan vara inblandade i den av utvecklarna upplevda användbarheten. Det förekommer motstridiga intressen mellan teorin bakom XP och utvecklarna. Tillämpningen av metoden enligt Becks anvisningar möter förhinder. Det som teorin lovar leverera, räcker av någon anledning inte för att utvecklarna ska följa den i praktiken.

Resonemanget leder oss fram till följande fråga i vår undersökning: Hur upplever systemutvecklarna samt deras organisation XP-metodens användbarhet i praktiken?

Användbarhet avser om systemutvecklarna upplever XP-metoden som attraktiv och applicerbar i en systemutvecklingsmiljö för att leverera ett tillfredsställande resultat. De parter som är av intresse är både systemutvecklaren och organisationen i vilken systemutvecklingen sker. Systemutvecklarna är de som ingår i utvecklingsprojektet där XP tillämpas och organisationen är den i vilken projektet utförs. Organisationen innefattar bland annat ledning och företagskultur.

1.3 Syfte

Syftet är att belysa olika faktorer som utgör XP-metodens användbarhet och hur dessa kan upplevas i praktiken. Skillnader och likheter mellan det rekommenderade arbetssättet och den praktiska tillämpningen belyses ur ett användbarhetsperspektiv

En sökning efter akademiskt material på Elin@Lund med sökorden ”extreme programming” och ”usability” ger ett mycket torftigt resultat. Även vid djupare efterforskning kring upplevd användbarhet gällande XP-metoden är det problematiskt att finna ämnesträffande litteratur. Vi vill med denna uppsats leverera ett bidrag till ökad kunskap inom detta område.

Även om metoden har som syfte att generera en effektiv och produktiv utvecklingsprocess är det ingen självklarhet för att tillämpning av metoden är smärtfri.

Eftersom agila metoder i allmänhet verkar vara vanligt förekommande och den kunskap vi hoppas kunna tillföra kan vara ett viktigt verktyg vid bedömning av om, och i så fall hur, XP ska tillämpas (Larman & Basili, 2003). Relevant kunskap om hur en metod fungerar gällande tillvägagångssätt vid tillämpning och konsekvenserna av resultatet av nyttjandet, kan generera en tydligare och därmed tryggare bild av konsekvenserna vid tillämpning. En lyckad implementering av en metod som utlovar en effektivare arbetsprocess kan självfallet vara värdefullt för företag som önskar tillämpa det agila tankesättet. En bra fungerade metod, anpassad eller inte, kan leda till att projekt effektiviseras, fallgropar undviks och risken för misstag minimeras, vilket innebär att det finns ekonomisk vinning med att inhämta den kunskap vi hoppas kunna bidra med. Även kunskap om situationer eller aspekter som innebär en ofördelaktig miljö för XP-tillämpning är av värde, då beslutsunderlaget för metodvalet kan tas med högre precision och tillförlitlighet.

1.4 Avgränsning

Vi vill med denna uppsats inte svara på *när* XP ska tillämpas, utan istället belysa den av systemutvecklarna uppfattade användbarheten. Vidare gör vi ingen ansats att förklara *hur* en framgångsrik tillämpning av metoden skulle kunna genomföras.

Kent Beck har presenterat en uppdaterad version av XP, men det är inte den som uppsatsen grundar sig på. Uppsatsen baseras på den första av de två versionerna av XP, som gavs ut år 1999. Den andra versionen verkar ha fått ett begränsat genomslag och är framför allt riktad mot en större projektgrupp. (Beck & Andres, 2005)

2 Teoretiska utgångspunkter

I detta kapitel presenterar vi den teori som ligger till grund för våra antaganden. Kapitlet behandlar en översikt över Extreme Programming, användbarhet, systemutvecklingsmetoder i praktiken, XP i praktiken samt avslutas med vårt teoretiska ramverk som sammanfogar teoridelarna.

2.1 XP-metoden – en översikt

XP-metoden är grunden i denna uppsats och en översikt av metoden syftar inte bara till att presentera den. Det är nämligen lika viktigt att belysa det som föranledde metodens skapande, det vill säga vilka problem och svårigheter i systemutveckling som det var tänkt att metoden skulle hantera.

2.1.1 Bakgrund

Det bakomliggande tänk som XP grundar sig på är inte så nytt och unikt som det kanske kan ge sken av, utan tänket har sin grund mycket längre tillbaks i tiden än tidpunkten för XP-metodens lansering. Det iterativa och inkrementella arbetssättet (IIA) har förekommit sedan 1930-talet, då det applicerades inom tillverkningsindustrin för att höja kvaliteten på resultatet. På 1950-talet började IIA att tillämpas av NASA vid utvecklingen av deras farkost X-15 och IIA ansågs vara bidragande till projektets framgång. Detta ledde till att NASA även använde IIA vid utveckling av mjukvara i det påföljande Mercury-projektet. (Larman & Basili, 2003; Dick, 2000) IIA fortsatte att brukas inom mjukvaruutvecklingen, men det var först i slutet av 1990-talet som tillämpningen tog fart. Flera nya metoder med grund i IIA presenterades kring denna tidpunkt och år 2001 gick flera av deras skapare samman och deklarerade intressegruppen ”The Agile Alliance”. Gruppen förkunnade sin ståndpunkt genom att kungöra ”The Agile Manifesto”, som är ett manifest för det så kallade agila tillvägagångssättet. (se bilaga B6) Manifestet är undertecknat av flera skapare av agila metoder, bland annat Kent Beck, skaparen av XP. (Larman & Basili, 2003; Highsmith, 2001) Popularitеten för de agila metoderna vid denna tidpunkt ledde även till att föreningen ”Agile Sweden” startades år 2002. Syftet med föreningen var att studera det agila synsättet närmare för att kunna anpassa detta till svenska förutsättningar. (Agile Sweden - nätverket för flexibla systemutvecklingsmetoder, 2008)

Gemensamt för de agila utvecklingsmetodikerna är det nära samarbetet mellan utvecklarna och kunden, dokumentation byts mot kommunikation, frekvent leverans av affärsvärde genom en mängd releaser, små självorganiserade grupper av utvecklare samt möjligheter att utveckla kod som på ett oproblematiskt sätt hanterar förändringar i kravspecifikationen. (Agile Alliance: What Is Agile Software Development?, 2006)

Några andra agila utvecklingsmetoder är (Avison & Fitzgerald, 2006):

- Scrum
- Crystal
- Agile Modelling(AM)
- Adaptive Software Development (ASD)
- Feature Driven Development

Av dessa har framför allt Scrum blivit populär den senaste tiden. Metoden baseras på sprintar som varar mellan en till fyra veckor i vilken systemet byggs inkrementellt. Teamet leds av en Scrum master som är ansvarig för projektets backlog, de uppgifter som teamet skall utföra. Inför varje sprint gör Scrum mastern en planering av vad i backlogen som ska utföras. Dessutom är Scrum mastern ansvarig för uppföljning. Varje dags hålls ett Scrum meeting som är 15-30 minuter långt, under vilket man diskuterar vad varje utvecklare har åstadkommit sedan förra mötet, vilka problem denne har stött på samt vad som skall göras idag. (Rising & Janoff, 2000)

Lanseringen av XP

XP:s grundare Kent Beck har en tekniskt inriktad bakgrund med en mastersexamen i datavetenskap från University of Oregon i Oregon, USA . Han presenterade metoden år 1999 och den fick snabbt mycket uppmärksamhet och växte till att bli den mest kända agila metoden. (Abrahamsson, 2003; Karlström, 2002)

Metoden presenterades dels genom publiceringen av en artikel i tidsskriften Computer (Computer : a publication of the IEEE Computer Society, 2008) och dels genom boken, ”Extreme Programming explained – Embrace change”. (Amazon.co.uk, 2008). I artikeln beskrev Beck kortfattat de problem han upplevde fanns vid mjukvaruutveckling och lade fram XP-metoden som ett försök att lösa dessa (Beck, 1999). I boken redogörs metoden i detalj, från de grundläggande värderingarna till de tolv praktikerna. (Beck, 2000; Fruhling et al., 2008)

XP skapades alltså som ett försök att lösa de problem som fanns inom systemutveckling, nämligen att det var problematiskt att leverera mjukvara och framförallt att leverera den inom budget, i tid och med rätt kvalitet (Fitzgerald et al, 2006; Slaughter, Harter, & Krishnan, 1998). I en värld där kraven som ställs på mjukvara är vaga och ständigt förändras, hävdar Beck att det krävs en disciplinerad användning av så kallade ”best practices” för att lyckas. XP bygger på detta och han kallar dessa för *praktiker*. Alla praktiker är noggrant utvalda för att samverka genom att komplettera och understödja varandra. Allt för att skapa en så komplett metod som möjligt, där alla praktiker skall användas fullt ut. Eller som Beck uttrycker det, XP tar var praktik till en *extrem* nivå. (Beck, 2000)

Idén till XP fick Beck under ett systemutvecklingsprojekt på Chrysler. Beck plockades in utifrån för att styra upp det pågående projektet, då det drogs med stora bekymmer. Att lösa de problem som fanns var extra viktigt då företaget var beroende av det lönesystem som utvecklades. Beck strukturerade om arbetssättet genom att successivt introducera nya tillvägagångssätt, vilket senare kallades praktiker. Idéerna till dessa föddes under personliga möten med var och en av utvecklarna, där problem och möjliga lösningar diskuterades. Till slut var tillräckligt många idéer samlade och XP skapades. (Plamondon, 2003)

XP:s användningsområde

Det finns ett antal viktiga företeelser att ta i beaktning gällande XP, bland annat att metoden är tänkt att tillämpas på små utvecklingsprojekt bestående av två till tio användare. Vidare ska även XP ses som ett regelverk eftersom vissa förutsättningar ska uppfyllas för att det ska klassas att det är XP som utövas. Det finns alltså inga valmöjligheter med avseende på hur tillämpningen ska ske. I klarspråk betyder det att samtliga tekniker skall användas, annars används inte XP. (Beck, 2000)

2.1.2 XP-metodens uppbyggnad

I början av traditionella systemutvecklingsprojekt görs en insamling av de krav som systemet ska uppfylla och dessa krav ligger sedan till grund för det fortsatta arbetet. En av de mest klassiska systemutvecklingsmetoderna som använder sig av det här tillvägagångssättet är vattenfallsmetoden. Den följer några enkla steg som skall utföras i kronologisk ordning och varje steg skall vara avslutat innan nästa påbörjas. (Avison & Fitzgerald, 2006)

1. Kravanalys
2. Design
3. Implementering
4. Test

Detta tillvägagångssätt skapar bekymmer, bland annat därför att processen blir väldigt rigid. Väcks det ett intresse att lägga till en funktion, till exempel under implementeringsfasen, så uppstår genast problem. Tidigare steg måste därför helt eller delvis göras om, vilket kostar tid och pengar. I dagens snabbt föränderliga värld är det snarare regel än undantag att kravförändringar förekommer och det tidigare antagandet, att krav kunde slutgiltigt fastställas i början av projektet, har visat sig stämma dåligt överens med verkligheten. För att lösa denna problematik skapades ett antal agila systemutvecklingsmetoder. Deras syfte var att hantera de kravförändringar som sker efter det att design, implementering och test påbörjats. XP är en sådan metod. (Highsmith & Cockburn, 2001)

Grundstommen i XP kallas för iteration. Varje iteration inleds med att kunden väljer ut den funktionalitet som skall vara implementerad vid iterationens slut. Därefter följer faserna design, implementering och test. Ett helt systemutvecklingsprojekt innehåller att antal iterationer, där varje iteration oftast inte är längre än några veckor. Det grundar sig på XP:s filosofi att systemet alltid ska ha en konstant kvalitet och att det är systemets *omfattning* som skall variera med tiden. (Beck, 2000)

Detta resonemang bygger på järntriangeln. Tanken är att se kostnad, tid och kvalitet som delar av en triangel. När omfattningen är bestämd kan de tre delarna varieras för att uppnå resultatet. Exempelvis kan en försening undvikas genom att mer pengar skjuts till eller att kvaliteten sänks. (Atkinson, 1999; Beck, 2000; Chow & Cao, 2008)

- Kostnad – budgeten för projektet
- Tid – tidsramen för projektet
- Kvalitet – kvaliteten på resultatet
- Omfattning – funktionalitetens omfattning

Vid varje iterations slut finns körbar mjukvara, vilket innebär att det alltid finns en möjlighet att leverera ett affärsvärde till kunden trots att alla funktioner inte är implementerade. En

fördel med detta är att om pengarna eller tiden skulle ta slut kan projektet enkelt avbrytas, men trots detta har affärsvärde levererats. Genom att låta omfånget variera kan projekttiden, budgeten och kvaliteten hållas på en konstant nivå och det enda som eventuellt saknas vid projektets slut är funktionalitet. Ytterligare en fördel är att kravinsamlingen kan ske kontinuerligt under projektet innebär att systemet kan växa fram under förändrade förutsättningar. Vaga och otydliga krav kan dessutom tydliggöras genom att kunden får feedback när systemet växer fram och då bättre kan avgöra vad som önskas respektive inte önskas av systemet. Vid nästa iteration kan projektets riktning korrigeras och nya förslag och krav kan läggas till. (Beck, 2000)

2.1.3 Praktiker

De tolv praktikerna

Det kan låta som en trivial uppgift att få en samling tillvägagångssätt presenterade för sig och sedan följa dem, men trots att metoden kan ge sken av att vara enkel att använda som det är tänkt, finns det flera bevis för det motsatta (Hedin et al., 2005; Mirakhorli et al., 2008).

De tolv praktikerna

I tabellen nedan görs en kort presentation av praktikerna.

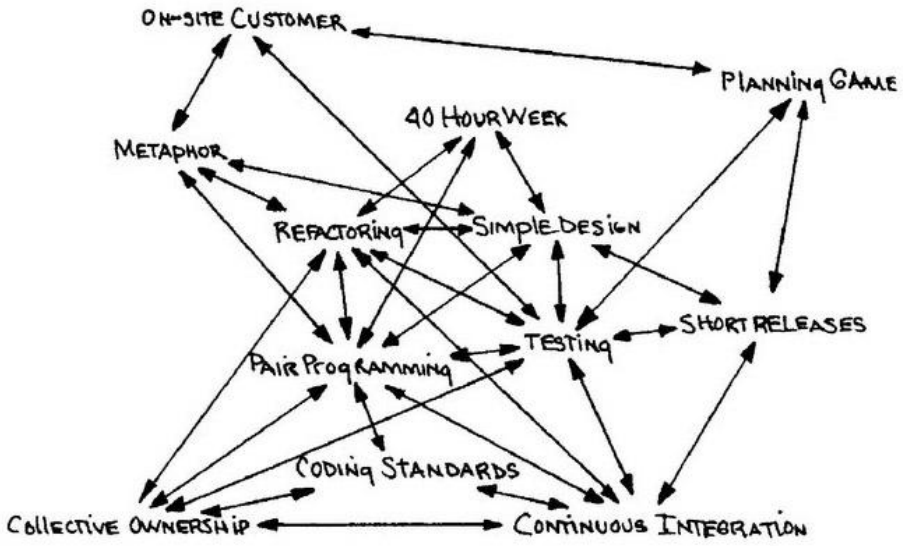
Tabell 2.1 - De tolv praktikerna i XP (Beck, 2000).

Praktik	Förklaring
The Planning Game	I början av varje iteration väljer kunden ut vilka funktioner som ska implementeras. De viktigaste funktionerna väljs ut till dess att tidsutrymmet för iterationen tar slut. Därefter gör utvecklarna en planering för iterationen.
Small releases	Vid varje ny release ska ny funktionalitet finnas med. Releaserna ska släppas ofta för att skapa ett ökat kundvärde.
Metaphor	För att utvecklingsmiljön skall bli enklare att förstå för alla inblandade, exempelvis kunden, används metaforer istället för tekniska termer. Skrivbordet i Windows skulle exempelvis kunna liknas vid ett vanligt skrivbord.
Simple Design	En enkel design syftar på en design som löser problemet med minsta möjliga komplexitet. Det handlar om att få systemet så enkelt och lättförståeligt som möjligt under de förutsättningar som ligger till grund. Med andra ord designas systemet bara utifrån de problem och krav som finns idag, inte med eventuellt kommande funktioner i åtanke.
Testing	Genom att skriva test skapas en trygghet i att funktionerna i programmet fungerar. Tryggheten gäller både för systemutvecklarna och för kunden. Testerna gör att utvecklarna kan ändra i koden med minskad risk för att inte upptäcka om något går snett.
Refactoring	För att hålla en konstant nivå på enkelheten i systemet behöver koden omstruktureras med jämna mellanrum. Vid adderandet av ny funktionalitet skall systemet modifieras för att även efter implementeringen vara i sin simplaste form. Risken är annars att komplexiteten ökar för varje ny funktion som läggs till.
Pair programming	Programmerarna sitter två och två vid varje dator. En skriver programkod och den andre sitter bredvid och granskar. Tack vare

	detta finns det alltid en extra uppsättning ögon som kontrollerar koden och kan upptäcka fel och förbättringsmöjligheter. Dessutom finns alltid någon att diskutera eventuella frågetecken med.
Collective code ownership	Så fort någon programmerare behöver ändra i koden skall personen göra så. Alla äger hela koden och innefattar lika ansvar för den.
Continous integration	Minst en gång om dagen bör programmerarna skicka in sin kod till huvudbiblioteket och kontrollera att allt passar ihop. Om detta inte görs tillräckligt ofta kan synkroniseringsproblem i arbetet uppstå.
40-hour week	En arbetsvecka är 40 timmar och denna gräns ska gälla under utvecklingen. Syftet med att arbeta utan övertid är att skapa en kreativ och idéfylld arbetsmiljö. Tanken är att då alla praktiker följs så sker arbetet redan så snabbt som det är möjligt.
On-site customer	Detta innebär att kunden sitter i samma rum som teamet. Personen kan svara på frågor, lösa konflikter och göra mindre prioriteringar. Kunden har även som uppgift att skriva funktionella tester på det som teamet levererar, för att försäkra sig om att kraven uppfylls. Beck hävdar att när en kundrepresentant är på plats så levereras mycket mer värde när han eller hon hjälper teamet, än om personen i fråga hade arbetat med sina vanliga arbetsuppgifter.
Coding standards	Koden ska skrivas på ett standardiserat sätt som alla ska följa. Det underlättar kommunikationen och snabbar upp arbetet.

Praktikernas sammankoppling

Anledningen till att alla praktiker skall följas fullt ut beror på att de understödjer och kompletterar varandra. Om någon av praktikerna skulle avvaras innebär det inte bara att den praktikens fördelar försvinner, men också att andra praktiker undergrävs. Ta ”On-site customer” som exempel. Här kan det noteras att det finns pilar till ”Metaphor”, ”Testing” och ”The Planing Game”. En metafor underlättar för kunden att förstå frågor och problem av mer teknisk karaktär i systemet. Det kan vara till användning i planeringsspelet, där kunden är en viktig del för att välja ut de viktigaste funktionerna. Slutligen ger test en viktig feedback till kunden gällande hur projektet framskrider. (Beck, 2000)



Figur 2.1 - De tolv praktikerna i XP och hur de understödjer varandra. (Beck, 2000)

2.2 Användbarhet

Efter att ha fått en bild av vad XP-metoden är vill vi gå över till hur användbart XP blir i praktiken. För att kunna göra detta behövs först en konkret definition på vad användbarhet innebär. Det finns ett antal olika definitioner på användbarhet och vi har valt att använda oss av en definition som är lämplig att applicera på systemutvecklingsmetoder (van Welie et al., 2001). Den litteratur inom området som vi har letat efter har gällt användbarhet kopplat till antingen metoder eller produkter. Detta gör att ordet ”system” kan kopplas till både en produkt samt en metod. Anledningen är att båda har som syfte att bistå användaren att utföra en viss uppgift eller handling. Denna koppling innebär att XP-metodens föreskriva arbetssätt, det vill säga dess praktiker, ska jämställas med det tillvägagångssätt med vilket det är tänkt att en produkt skall användas. (Bevan et al., 1991)

Användbarhet – Ett svårdefinierat begrepp

Det finns ingen allmängiltig definition på vad användbarhet innebär och det kan heller inte uttryckas genom något objektivt mätvärde (van Welie et al., 2001). Detta trots att användbarhet ofta anses vara en viktig komponent vid utformning av olika system. (Boivie et al., 2006; Seffah et al., 2006)

Ordet användbarhet skapades i början av 80-talet i ett försök att ersätta ordet användarvänlig. Termen användarvänlig hade associerats med flertalet olika definitioner vilket skapade en otydlighet om vad termen verkligen innebar. Dessutom dök det upp flera alternativa betydelser som ökade otydligheten än mer. Tyvärr drabbade detta fenomen även termen användbarhet så småningom. (Bevan et al, 1991). Att använda termen användarvänlig eller lättanvänd kan sägas vara kopplat till hur enkelt ett system är att använda, men det säger ingenting om hur väl systemet faktiskt utför det användaren vill. Användbarhet är alltså ett uttryck som inbegriper mer än vad de tidigare termerna gör. De nya delarna är hur lätt det är att lära sig systemet, hur snabbt användaren kan nå sina mål, felfrekvens samt hur nöjd användaren är med att använda systemet. (Hartson, 1998) Flera författare, bland annat van Welie et al (2001) och Bevan et al., (1991) tar upp de här delarna som beståndsdelar i användbarhet. En generell definition används av Bevan et al., (1991) och är här översatt till svenska:

> *”Användbarhet definieras som enkelhet vid tillämpning samt acceptans av produkten av en viss grupp användare, som utför en viss typ av uppgift i en viss typ av kontext.”*
> (Bevan et al., 1991, s. 1)

Här syftar enkelhet till om en produkt kan användas av användaren när denne skall utföra en viss uppgift. Acceptans innefattar huruvida produkten kommer att användas och dessutom på vilket sätt användningen sker. Lättanvändhet i en viss kontext innebär användarens prestation och tillfredsställelse av själva tillämpningen och slutligen belyser kontexten de tre delarna - användaren, uppgiften samt den sociala och fysiska omgivningen. (Bevan et al., 1991)

Mäta användbarhet

Att kunna mäta användbarhet är viktigt för att ha möjlighet att göra bedömningar huruvida ett system har lyckats med att vara användbart. För att avgöra detta krävs ett antal mätparametrar (Hartson, 1998). Användbarhet mäts två delar enligt Bevan et al., (1991), det ena är prestationen och det andra är användarens uppfattning av produkten, exempelvis acceptans. För att mäta prestationen används fyra olika mätskalor, vilka presenteras nedan. (Bevan et al., 1991)

- Måluppnåelse (goal achievement) – Lyckades användaren lösa uppgiften och/eller hur nära var han eller hon målet.
- Arbetshastighet (work rate) – Produktivitet och prestationsförmåga i arbetet.
- Kunskapsinhämtning (knowledge acquisition) – Hur väl det går att lära sig systemet och med vilken inlärningshastighet som detta sker.
- Användning (operability) – Felfrekvens samt på vilket sätt funktionerna används.

Användarens acceptans och generella uppfattning av produkten är av betydelse för det fortsatta användandet. En användare som tycker om produkten är mer benägen till fortsatt användning än någon som inte tycker om den. Dock är lättanvändhet inte tillräckligt för att skapa acceptans utan användaren behöver också uppleva att produkten är effektiv och är användbar i arbetet. (Thong et al., 2004)

Konstruera användbarhet

Då användbarhet är beroende av användaren, uppgiften och miljön är det svårt att i förväg uppskatta vilka kriterier som är viktiga att ta i beaktning vid systemets konstruktion. Särskilt utmanande blir det då det nya systemet behöver var minst lika effektivt och tillfredsställande som den metod eller tillvägagångssätt det ämnar ersätta. Forskning sker löpande för att finna metoder som gör det möjligt att i förväg kunna avgöra vad som är viktigt för god användbarheten för ett specifikt, ännu inte utvecklat system. Än så länge får dock professionella bedömningar ligga till grund. Att presentera i förväg exakt hur ett system skall konstrueras är ingen trivial sak, vilket dessutom försvåras av att en liten förändring av användaren, kontexten, systemets utformning eller uppgiften kan förändra den upplevda användbarheten. (Bevan et al, 1991)

Det finns dock alternativa vägar att gå i jämförelse med att försöka använda sig av uträknade beskrivningar för vad som ger lyckad användbarhet i ett system. Det är att varje individ som deltar i designen av system ackumulerar färdigheter och erfarenheter under projektet, för att kunna använda sig av denna kunskap i framtida projekt. Dock är det inte sagt att utvecklare inte kan ta hjälp av designkunskap i form av exempelvis litteratur, men då tillgången till denna är begränsad har det visat sig att personlig erfarenhet har spelat en mycket större roll vid skapandet av framgångsrik användbarhet. Ofta är riktlinjer det enda konkreta hjälpmedlet som erbjuds och även om designern försöker använda dem, innebär det inte att det per automatik sker problemfritt. (van Welie et al., 2001)

Utvärdera användbarhet

Att finna en bra utvärderingsform för användbarhet är svårt (Bevan et al., 1991). Ofta görs utvärdering av användbarhet när systemet är konstruerat genom exempelvis olika test som användaren får utföra (van Welie et al., 2001). Det kan förekomma att användare i kontrollerade laboratoriemiljöer använder ett system för att lösa en uppgift och detta har visat sig fungera strålande. När sedan samma system används i verkligheten har det dock förkastats av användarna. Det är av denna anledning som *acceptans* är ett viktigt begrepp. För att användaren ska vilja använda systemet, ska hon eller han bedöma att fördelarna med att använda systemet är större än de alternativa metoder som finns tillgängliga för att lösa uppgiften. Användaracceptans grundar sig på karaktärsdrag hos användaren och även den kontext som användaren befinner sig i. Faktorer som kan påverka är bland annat kostnad, bekvämlighet, tillgänglighet, krävd träning och organisationsbegränsningar. (Bevan et al., 1991)

Van Weile et al., (2001) hävdar att då system är designade för människor är det viktigt att känna till deras möjligheter och begräsningar. Vidare spelar den sociala och organisatoriska delen roll. Generellt kan då sägas att användbarhetsproblem i grunden beror på en missanpassning mellan de färdigheter som användaren besitter och de färdigheter som systemet kräver av användaren.

2.3 Systemutvecklingsmetoder i praktiken

Men hur ser det ut när systemutvecklingsmetoder används i praktiken? Nedan presenteras en genomgång av hur användandet kan se ut och framförallt vilka faktorer som anses påverka när metoderna inte används så som det är tänkt. Förståelse för hur metoder används i praktiken är viktigt för att kunna utvärdera hur en metods användbarhet upplevs i praktiken, då detta kan skilja sig från vad metodens föreskriva arbetssätt ger sken av.

Beståndsdelarna som utgör en systemutvecklingsmetod är sällan revolutionerade inom mjukvaruindustrin utan istället är det ofta själva sammansättningen av delarna som definierar metoden. Detta har lett till en hel del diskussioner angående hur systemutvecklingsmetoder ska användas i praktiken. (Fitzgerald et al., 2006) Vissa hävdar att det är fritt fram att välja delar av metoder som upplevs passande, samt förkasta andra. Andra hävdar istället att metoder ska användas till fullo då de levererar en synergieffekt, vilket medför att delarna inte kan väljas på ett godtyckligt sätt. (Hussain et al., 2008; Luck, 2004; Beck, 1999) Andra upplevda problem är att metoderna verkar fokusera på vissa delar i systemutvecklingens livscykel, till exempel implementeringsfasen, samt bristande stöd för projektstyrning. De olika principer som utvecklarna ska rätta sig efter upplevs också ofta som abstrakta och svåra att följa. Det är inte ovanligt att en metod framställs som en universallösning som skall passa till allt, men som i praktiken inte har levererat bevis för detta i någon vidare omfattning. (Abrahamsson et al., 2003) Abrahamsson et al., (2003) föreslår att metoderna borde specialisera sig mer, istället för att försöka vara generella. Dessutom måste projektstyrningsbiten bli större för att fungera i organisationer. Praktikerna behöver även bli mer konkreta, så att de blir enklare att tillämpa samt för att möjliggöra bättre empirisk utvärdering. (Abrahamsson et al., 2003)

Att metoder inte upplevs helt förankrade i verkligheten har lett till en våg av metodanpassningar inom branschen. Fokus har legat på den unika kontext i vilken systemutvecklingen sker och vad som passar in i just denna. Exempelvis kan organisationsstrukturen i kombination med distribuerade team kräva förändringar i en existerande metod. (Fitzgerald et al., 2006)

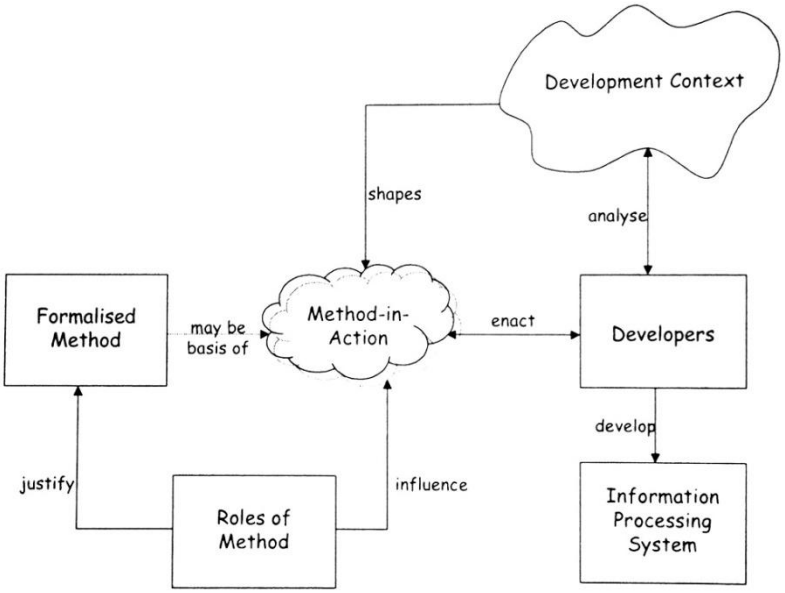
Två allmänt accepterade sätt att kategorisera metodanpassning är genom metodkonstruktion (method engieneering) eller faktorer i den nuvarande situationen (contingency factors). Gällande metodkonstruktion skapas en metod utifrån existerande metodfragment, till exempel praktiker, relaterat till förutsättningarna och behoven som finns. När det gäller faktorer i den nuvarande situationen så syftar det till att organisationen väljer ut vilken metod som ska användas baserat på hur situationen ser ut för det som ska utvecklas, där ett antal faktorer påverkar beslutet. Problemet som kan uppstå i och med detta är att det krävs att organisationen har djup kunskap om ett antal systemutvecklingsmetoder för att kunna ta ett bra beslut. (Bajec et al., 2007)

Metodanpassning är speciellt av intresse för företag som är angelägna av att ha en process som passar just dem. Att finna bevis för att XP fungerar i en kommersiell kontext är därför av

intresse, speciellt då Fitzgerald et al., (2006) menar på att många framgångsberättelser kring XP härstammar från den akademiska värden.

2.3.1 Ramverk för metoder i praktiken

Fitzgerald et al., (2002) har med bakgrund av en stor mängd litteratur samt eget insamlat empiriskt material skapat ett ramverk för hur utveckling av informationssystem ser ut i praktiken. De olika delarna i ramverket presenteras nedan:



Figur 2.2 - Ramverk för metoder i praktiken. (Fitzgerald et al., 2002)

Formaliserad metod

I slutet av 1960-talet utvecklades ofta system på ett godtyckligt sätt utan hjälp av formaliserade, nerskrivna metoder. Tillvägagångssättet var osystematiskt och ledde till en mjukvarukris. Denna kris innebar att utvecklingen av system överskred budget, drog över tiden samt levererades med undermålig kvalitet. I början av 70-talet lanserades idén att systemutvecklingsmetoder behövde bli mer formaliserade. Tanken var att skapa en rationell och vetenskaplig process. Detta är en del av de tankegångar som ligger till grund för vattenfallsmetoden. (Fitzgerald et al., 2002)

Formaliserad metod inbegriper både vetenskapliga och kommersiellt använda metoder. Det finns flertalet exempel på detta så som tidigare nämnda vattenfallsmetoden, men även RUP, RAD och XP. (Fitzgerald et al., 2002)

Metoder i verkligheten

I verkligenheten är systemutvecklingsmetoder sällan applicerade så som metodförfattarna tänkt sig bruket av metoden. Ofta ligger metoden istället till grund för utvecklingsarbetet. Utvecklarna tolkar metoden olika och själva tillämpningen skiljer sig även åt. Fitzgerald et al menar att varje utvecklare i ett projekt *agerar* metoden på ett unikt sätt. (Fitzgerald et al, 2002)

Systemutvecklingsmetodens roll

Det finns två olika typer av roller som en systemutvecklingsmetod kan spela i ett utvecklingsprojekt. Det första är en rationell roll som har sin bakgrund i ett rationellt resonemang där tillämpningen av metoden motiveras på ett tydlig och konkret sätt, till exempel genom att erbjuda verktyg för att hantera komplexitet. Den andra rollen är den politiska rollen, där en metod exempelvis kan utstråla trygghet för de inblandade. Den skapar en känsla av att följa något som är korrekt och som man som utvecklare kan luta sig emot. Vidare kan bruket av metoder fungera som konkurrensfördel och vara certifieringsgrundande. (Fitzgerald et al., 2002)

Utvecklingskontext

Systemutveckling sker alltid i en unik kontext. Affärskontexten är både komplex och dynamisk till sin natur vilket gör att systemutveckling alltid sker i en unik kontext. Fitzgerald et al., (2002) hävdar att en medvetenhet om denna unikhet är ett krav för att en systemutvecklingsmetod skall vara framgångsrik. Ofta är metoder konstruerade så att de ska fungera oavsett i vilken kontext som användningen sker. Detta dementerar dock Fitzgerald et al, som istället menar att den unika situation i vilken utvecklingen sker formar den och skapar på så sätt *metoder i praktiken*. (Fitzgerald et al., 2002)

Utvecklarna

Det är människor, inte metoder, som utvecklar system och därför är utvecklaren en mycket viktigt del av systemutvecklingen. Metoder är mer som ett ramverk, vilket skall balanseras med utvecklarnas kunskap för att bli effektivt. Ofta tar metoder exempelvis ingen hänsyn till utvecklarnas olika kunskaper och erfarenheter. Det finns mycket som talar för att utvecklarnas olika förutsättningar har stor påverkan på produktiviteten i projektet, dessutom ökar deras kunskap allt eftersom projektet fortskrider, vilket ytterligare bidrar till komplexiteten. (Fitzgerald et al., 2002)

Informationssystemet

Utvecklingen av informationssystem har tidigare varit väldigt fokuserat kring de tekniska aspekterna av arbetet, men Fitzgerald et al., (2002) hävdar att den sociala aspekten är lika viktig. Alla delar i en utvecklingskontext påverkar varandra. Kontexten hanteras olika beroende på uppgiften. Exempelvis är det stor skillnad när ett säkerhetskritiskt system utvecklas i jämförelse med ett generiskt lågrisksystem.

2.4 XP i praktiken

Efter att ha presenterat material som visar på att alla systemutvecklingsmetoder inte alltid används som det är tänkt fördjupar vi oss nu i hur det ser ut för XP på detta område.

2.4.1 Grad av tillämpning

Att använda XP kan leda till fördelar i jämförelse med andra systemutvecklingsmetoder i form av lägre kostnader, högre produktivitet, tidig konkret feedback samt högre kvalitet på resultatet (Mirakhorli et al., 2008; Fitzgerald et al., 2005). Det har framkommit ett antal nackdelar. Bland annat att det finns begränsningar gällande storleken på XP-projekt. Vidare

krävs en viss erfarenhetsnivå hos projektmedarbetare samt fysisk närvaro från kundens sida, men trots detta menar Mirakhorli et al., (2008) att XP är en av de mest populära agila metoderna. De begräsningar som nyss nämndes innebär att det inte är alla mjukvaruprojekt som kan dra nytta av XP, men framförallt kan de inte använda alla de praktiker som XP kräver, något som medför att metoden skräddarsys efter omständigheterna. (Mirakhorli et al., 2008; Fitzgerald et al., 2005).

Idag är världen turbulent och osäker vilket medför att de traditionella metoderna får det svårare att vara framgångsrika, då de har det besvärligare med att hantera kravförändringar. De agila metoderna är däremot tänkta att kunna hantera dagens organisationer och teknologiska miljöer. (Highsmith & Cockburn, 2001) Trots detta är det vanligare att XP anpassas än att metoden används i sin helhet. (Fitzgerald et al, 2006) De tolv praktikerna ska stödja varandra då XP som metod förutsätter att alla praktiker används, eftersom att de är stramt sammanbundna. Trots att exempelvis Schwaber & Beedle (2002) hävdar att denna sammansatta process inte är delbar och valbar finns det författare som hävdar att praktikerna i XP är en källa till problem.(Fitzgerald et al, 2006; Lindvall et al, 2004) En möjlig förklaring till detta kan vara att det är svårare att se förhållandet mellan praktikerna, än att se praktikerna själva. (Mirakhorli et al., 2008) Mirakhorli et al., 2008 hävdar att om de som tillämpar XP förstår förhållandet mellan praktikerna väl, så är det kanske möjligt att göra ”korrekta” ändringar i XP där det bakomliggande tänket är tämligen intakt.

Framförallt två problem uppstår på grund av att XP definieras av sina praktiker. Det första gäller att några av dem är oklara i sin definition, det är svårt att konkretisera vad som menas. Det andra handlar om att det är en vanligt förekommande ”sanning” att XP kan användas utan att alla praktiker används. (Lindvall et al., 2004; Mirakhorli et al., 2008) Att summan är större än delarna när en metoden används motbevisas av några studier. Några ser XP som en *lättviktsmetod,* där man kan plocka delar för att komplettera den befintliga agila metod man använder och nämner i detta sammanhang att XP inte är känt för att vara en ”one size fits all”-metod som passar alla utvecklingskontexter. Samtidigt påpekas att många av XP:s styrkor har framkommit genom studier gjorda i akademiska miljöer och menar därför att praktiskt gjorda studier i arbetslivet är av stor vikt. (Fitzgerald et al., 2005)

XP som komplement till andra metoder

Fitzgerald et al. 2005 utförde en studie på Intel Shannon i Irland där de utvecklar mjukvara och kiseldesign med 125 anställda, där 90 av dessa är involverade i utvecklingen. Deras studie visade på att agila metoder i allmänhet inte klarade av att ensamma stödja utvecklingsprocessen, men att XP och någon systemutvecklingsmetod som innefattar mer projektstyrning, som Scrum, kompletterar varandra väl. XP ger stöd för den tekniska biten av projektet medan Scrum ger stöd åt projektplaneringen. Deras studie visar också på att i allmänhet är en blandning av metoder och en anpassning av desamma en lyckad strategi. (Fitzgerald et al., 2005).

2.4.2 Användning av praktikerna

Nedan följer en kort genomgång av hur praktikerna har upplevts i praktiken och fokus ligger på att belysa hur de har fungerat att använda. Vissa praktiker har ett visat på ett mycket bra utfall medan andra visar på ett fåtal framgångar. Slutligen presenteras de praktiker som har visat både bra och dåliga sidor.

”The planning game” ger bra överblick över arbetet vilket bidrar till att göra nya bedömningar under projektets gång, till exempel om en funktion behöver läggas till. Det har visat sig vanligt att detta sker då kunden från början inte alltid har helt klart för sig hur systemet skall se ut. (Karlström, 2002; Murru et al., 2003; Fruhling et al., 2008) Det är dock viktigt att alla i teamet får ta plats och vara delaktiga, annars finns risken att endast några, framför allt seniora utvecklare, tar för mycket plats med avseende på att göra uppskattning och föreslå lösningar. (Murru et al., 2003) I en studie gjord av Hedin et al., (2003) bestående av mest juniora utvecklare upplevdes som svårt att planera hur mycket som skulle få plats i en iteration och ibland gav utvecklarna upp och istället programmerade så mycket de hann. (Hedin, Bendix, & Magnusson, 2003) Trots att det finns positiva erfarenheter av ”The planing game” är den även en av de praktiker som upplevs vara mest problematisk att tillämpa, exempelvis om kunden inte finns tillgänglig i rummet, i form av ”customer-on-site”, för att kunna göra bedömningar och prioriteringar. (Rumpe & Schröder, 2002) Att inkludera kunden i projektet är viktigt för att ”The planing game” skall fungera (Schuh, 2001). Dock har det visat sig vara svårt att övertala kunden om nyttan med att ha en representant fysiskt närvarande hos utvecklarna, vilket har lett till att detta är en av de praktikerna som upplevs vara mest problematisk att uppfylla. (Rumpe & Schröder, 2002) Viktigt i sammanhanget är dock att i de fall som kunden har haft en representant närvarande så har resultatet fallit väl ut. (Karlström, 2002; Hussain et al., 2008)

En annan praktik som spelar en viktig roll i ”The planning game” är ”Metaphor”. Det har visat sig vara svårt att tillämpa metaforer i praktiken på grund av de svårigheter med att hitta en metafor som passar. (Karlström, 2002; Hussain et al, 2008) I en undersökning av Rumpe et al., (2002) görs en sammanställning över vilka praktiker i XP som är mest problematiska och där återfinns ”Metaphor” i topp. I hela 68,9% av de undersökta projekten ansågs metafor vara en problematisk praktik och anledningen ansågs primärt vara att det var svårt att förstå hur den skulle tillämpas. (Rumpe & Schröder, 2002)

Att följa ”40-hour week” kräver en effektiv organisation med välplanerade möten och schema över hur implementeringen ska ske. Denna praktik kan även ses som en ”mänsklig” del av XP. (Sfetsos et al., 2006)

En praktik som däremot var lätt att förstå och oproblematisk att ta till sig var ”small releases” (Rumpe & Schröder, 2002). Den ses som klart framgångsrik och en av de viktigaste praktikerna för att nå framgång i ett XP-projekt. Anledningen är att den bland annat ger viktig feedback från användare och kunder (Karlström, 2002; Fruhling et al., 2008; Hussain et al., 2008). Det bekräftas också av att ”small releases” är en av de praktiker som få anser innebär problem. För att frekvent vara kapabel att släppa en uppdaterad version av systemet krävs omfattande tester för att förvissa sig om att det verkligen fungerar, annars kan inte en körbar version levereras. Men att använda ”test first” är inte helt bekymmersfritt. Det har visat sig vara svårt för många att veta på vilken nivå testerna skulle läggas på. Var testerna alldeles för kodspecifika fick testerna skrivas om för minsta lilla ändring i koden. Om testerna istället var satta på en för hög nivå missade de att fånga småfel. Det kan således vara svårt att avgöra vilka test som skall skrivas till en viss funktionalitet. Praktiken behöver exemplifieras för utvecklarna, då detta underlättar för dem att sätta sig in i den. Svårigheterna ligger framför allt i att test skall skrivas innan man skriver den kod som testet skall kontrollera, vilket är tvärt emot det traditionella sättet. (Hedin et al., 2003; Bendix et al., 2005; Karlström, 2002) En fördel med ”test first” är dock att testerna som utförs pekar ut den riktning som behöver tas för att de skall ha möjlighet att lösas. Fokus flyttas således från att skriva kod som skapar funktionen till att få testerna att passera. (Fitzgerald et al., 2006) Samtidigt har det visat sig att

”test first” var en praktik som ibland upphörde att tillämpas, framförallt i stressade situationer då de kortsiktiga vinsterna kunde kännas otydliga (Karlström, 2002)

Test har visat sig vara viktigt för flertalet praktiker, bland annat för ”refactoring”, som är en praktik med gott rykte då den underlättar eliminerandet av buggar. Framför allt underlättar den elimineringen i början av projektet, då problem som hade uppkommit då hade tagit betydligt mer tid i anspråk att lösa senare i projektet. (Fitzgerald et al., 2006) Dock kan det innebära problem då det krävs mod för att våga utföra en större omstrukturering av koden, eftersom att det kan påverka hela gruppen. Praktiken kräver även att utvecklaren upptäcker när ”simple design” inte längre är gällande och således att en omstrukturering krävs. (Hedin et al., 2005) ”Refactoring” är dock en praktik som trots att den syftar till att höja produktiviteten på sikt genom att eftersträva enkel kod även kan uppfattas som problematisk vid produktivitetsuppskattning. Att strukturera om kod kan innebära mycket arbete som varken genererar ökad funktionalitet eller ett ökat antal rader kod. (Abrahamsson, 2003) Det har dock visat sig att ”simple design” är viktigt i arbetet då tid har sparats genom att möjligheten finns att undvika onödigt omfattande lösningar samt tidsbesparing, då enklare kod har varit snabbare att hantera. (Karlström, 2002; Fruhling et al., 2008; Jackson et al., 2004) Praktiken ”continous integration” är även den beroende av test. Att implementera kod i det gemensamma biblioteket utan att förvissa sig om att detta fungerar skulle kunna bli problematiskt. ”Continous integration” har även visat sig vara framgångsrik samt enkel att tillämpa och även bidrar till att hålla nere synkroniseringskonflikter. (Bendix et al, 2005; Karlström, 2002; Rumpe & Schröder, 2002) Kopplat till detta finns även ”collective ownership” som få anser sig finna problem med. Däremot handlar det om modighet, då det kan finnas rädsla för att ändra i koden. (Rumpe & Schröder, 2002; Karlström, 2002)

Att använda parprogrammering har visat sig medföra en låg felfrekvens på den producerade koden, vilket i sin tur har inneburit högre kvalitet. Ytterligare en fördel var att risken för att paren körde fast var klart lägre än när programmerarna satt ensamma. (Karlström, 2002; Fitzgerald et al, 2006) Parprogrammering bidrog även till att kunskapen i teamen spreds bättre samt skapade en djupare förståelse över systemet och dess design. (Fitzgerald et al, 2006) Beck menade att ”coding standards” skulle underlätta parprogrammering, men detta har i praktiken visat sig vara något som teamen slarvar med att nyttja. (Karlström, 2002)

2.4.3 Systemutvecklingens kontext

Den kontext i vilken systemutveckling sker påverkar utvecklingen på olika sätt och är därför av intresse när metodanvändning skall utredas. (Williams et al., 2004) Att exakt definiera vad som menas med kontext är ingen trivial uppgift. Flera författare nämner kontexten som en viktig faktor vid systemutveckling, men det är samtidigt inte alltid tydligt vad som i själva verket menas med kontext. Exempelvis tar Layman et al., (2004) upp kontexten som en viktig faktor att ta i beaktning. Även om XP har ansetts framgångsrik i en studie där kontexten ansågs spela roll, är det inte säkert att kontexten definierades djupare. (Layman et al., 2004)

Det ligger i vårt intresse att titta närmare på vad kontexten innebär, samt göra ett försök att konkretisera begreppet. Fitzgerald et al., (2002) beskriver kontexten som:

> *”We define the context as both the place where the information system will be implemented and the environment within which the development process will take place”.*
> (Fitzgerald et al., 2002, s. 110)

Vidare behandlar Fitzgerald et al., (2002) möjliga faktorer att beakta i kontexten så som teknologi och kultur, men fokuserar framförallt på att olika kontexter innebär olika benägenhet för förändring. Då flera författare har denna typ av vaga beskrivning eftersöker vi en mer konkret definition som kan vara mätbar för att få fram användbarhet.

Jones (2000) har studerat ett stort antal faktorer som påverkar projektet i den kontext det befinner sig, närmare bestämt nämner han 250 stycken sådana faktorer. Dock menar han på att det i de flesta projekt snarare rör sig om 10-20 olika större faktorer som kan kategoriseras i sex olika kategorier. Williams et al., 2004 använde sig av detta för att skapa ett ramverk för utvärdering av hur väl XP tillämpas av en organisation, samt resultatet av denna tillämpning. Williams et al:s (2004) ramverk bygger på Jones resonemang men är inriktat mot just XP. Exempelvis adderas en sjunde kategori till ramverket. Vissa faktorer behandlas kortfattat i tabellen nedan medan de större, som denna uppsats fokuserar på, presenteras i separata delkapitel. De faktorer som påverkar i kontexten är följande:

Tabell 2.2 - Kontextens olika beståndsdelar enligt Willians et al., (2004)

Software classification	Vad är det för typ av mjukvara som utvecklas? Läs mer i kapitel 2.4.6
Sociological factors	Olika typer av sociala faktorer som inriktar in sig på den enskilda utvecklarens egenskaper men även teamets sammansättning. Läs mer i kapitel 2.4.4
Project-specific factors	Hur ser projektet ut? Behandlar storlek, kostnad, kvalitetsfokusering med mera. Läs mer i kapitel 2.4.6
Ergonomic factors	Kontorslandskapet spelar en betydande roll när det gäller kommunikationen mellan utvecklare och kunder, vilket medför att XP passar bäst i kontorsmiljö som tillämpar öppet landskap. Att arbeta i enskilda rum hindrar kommunikationen vilken då sker mest genom cheferna. Dessutom påverkar miljön tillämpningen av samarbetspraktiker, till exempel parprogrammering. Miljön måste möjliggöra för detta samtidigt som ljudnivån och distraktionerna måste beaktas.
Technological factors	Olika team använder olika typer av metoder och tekniker för att underlätta deras arbete. Det kan exempelvis vara rigorös projektstyrning eller ett versionshanteringssystem för programkoden.
Geographic factors	Hur teamet sitter och arbetar. Behandlar till exempel distribuerade team och olika former av kundnärvaro. Belyses närmre i kapitel 2.4.5
Developmental factors	Behandlar vilken typ av utveckling som passar bäst för ändamålet. Krävs det att processen hanterar frekventa kravförändringar och leveranser eller kan en traditionell utvecklingsmetod räcka. Detta påverkas också av storleken på teamet samt i vilken grad utvecklarna är benägna till ha stort eget ansvar och arbeta i en fri, ”kaosartad” miljö. Läs mer i kapitel 2.4.4 och kapitel 2.4.6

2.4.4 Systemutvecklarna och teamet

Att systemutvecklarna skulle vara en av de viktigaste komponenterna i utvecklingskontexten är kanske inte så främmande. Det är viktigt att komma ihåg att det är människor och inte metoder som utvecklar system. Metodernas funktion är på sin höjd tänkt att fungera som hjälp till utvecklarna genom att strukturera arbetet. (Fitzgerald, 2002)

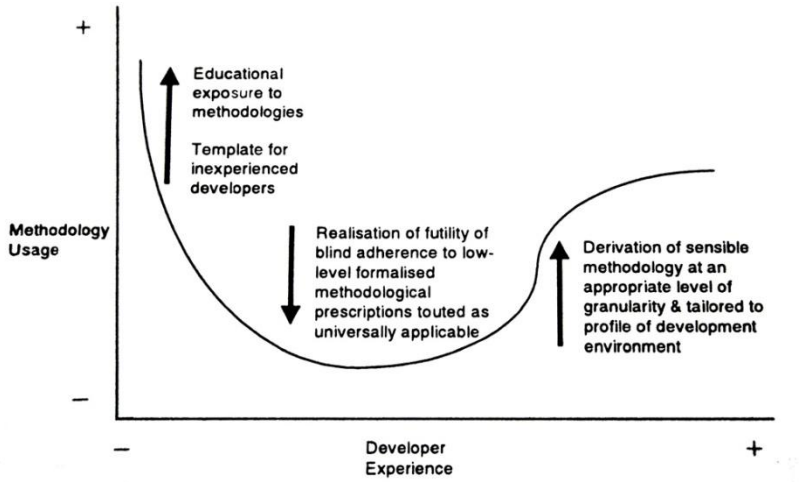
Williams et al., (2004) menar att storleken på teamet, utvecklarnas erfarenhet och personalomsättningen i teamet har stor påverkan på projektets framgång. Ökar storleken på temat ökar svårigheterna med kommunikationen och komplexiteten växer. (Muller & Tichy, 2001) Beck (2000) menar därför att XP inte är lämpat för större grupper av utvecklare eftersom kommunikationen är en sådan central del i metoden. Det är dessutom viktigt att sammansättningen av personerna i teamet fungerar. Att teamet innefattar utvecklare som har specialiserat sig inom något område kan påskynda processen då de kan fokusera vid sina kärnområden (till exempel GUI eller säkerhet). De kan även bistå med att sprida direkt kunskap till andra i gruppen. Vidare kan en deltagare i en mer ledande roll med erfarenhet av projektstyrning bidra till att hjälpa teamet med planering, kostnadsuppskattning och konfliktlösning. (Williams et al., 2004)

Williams et al., (2004) nämner personalomsättning som en viktig, påverkande faktor i ett XP-team. Då XP förespråkar minimalt med dokumentation leder detta till att den största delen av kunskapen finns hos projektmedlemmarna (Beck, 2000). Det är därför viktigt att kunskapen sprids, så att eventuella avhopp inte får en allt för stor negativ effekt på projektet (Williams, et al., 2004) Beck (2000) menar att detta fenomen kan förebyggas genom omfattande kommunikation samt användandet av metaforer inom teamet. Trots detta menar Williams et al., (2004) att XP gör det svårare för teamet att hantera eventuella avhopp från utvecklare.

För att lyckas som systemutvecklare, menar Fitzgerald et al., (2002) att finns det flera viktiga egenskaper en sådan skall besitta. Bland dessa återfinns att vara designer, ha kommunikationsfärdigheter, kunna förställa sig resultat och visa omdöme. Dock innebär inte detta att all utveckling sker på ett metodiskt sätt. Utvecklare antas ofta vara rationella, men så är inte alltid fallet. (Fitzgerald et al., 2002) Alla systemutvecklare trivs inte med att arbeta i en mer kaosartad miljö, som en agil metod innebär, utan trivs bättre under mer strukturerade förhållanden. (Williams et al., 2004)

Lawrence (2007) vittnar om att bristande erfarenhet hos juniora utvecklare kan leda till att arkitekturen i systemet inte växer fram på samma sätt som den hade gjort med seniora utvecklare. Juniora utvecklare har inte den grundkunskap som låter dem göra välgrundade tekniska beslut gällande design, vilket resulterar i dålig kod. Att enbart lämna över ansvaret till utvecklarna, oavsett erfarenhet och låta dem följa praktikerna räcker inte. (Lawrence, 2007)

Den erfarenhet som eftersträvas är den som byggs på, då antalet projekt som utvecklaren deltar i ökar. Under den här tiden byggs strategier upp mer eller mindre medvetet, för hur vissa typer av problem skall hantera och lösas. (Fitzgerald et al., 2002) Williams et al., (2004) instämmer och menar att den ackumulerade erfarenheten leder till en effektivare process med högre kvalitet. Dessutom sprids ofta denna kunskap aktivt i projektet till mindre erfarna projektdeltagare. Erfarenhet är dock inte allt, även en erfaren utvecklare kan stå handfallen när denna ställs inför ett nytt problem eller arbetar inom ett nytt område. (Williams et al., 2004) Samtidigt är det vanligare att erfarna utvecklare använder sig av systemutvecklingsmetoder i arbetet jämfört med dem som inte har lika mycket erfarenhet. Anledningen till detta är att de har lättare för att inse nyttan med dem. Dock finns det dem som hävdar att det istället är tvärt om, på grund av att erfarna utvecklare ser metoder som begränsande. Detta är något som Fitzgerald et al., (2002) har funnit i sin forskning. I början fungerar metoder som något som utvecklarna kan luta sig mot. Sedan ses de som hindrande. Därefter visar sig nackdelarna med att ingen metod följs, vilket resulterar i att metoden används igen, men på ett skräddarsytt sätt, genom att vissa delar väljs ut. Detta åskådliggörs i figur 2.4. (Fitzgerald et al., 2002)



Figur 2.3 – Metodanvändning i relation till utvecklarens erfarenhet. (Fitzgerald et al., 2002)

2.4.5 Utvecklarorganisationens perspektiv

Organisationer och systemutvecklingsmetoder

Formaliserade systemutvecklingsmetoder utvecklades som ett försök att hantera den enorma komplexitet som systemutveckling kan innebära. Metoderna skall hjälpa organisationen att hantera utvecklingsprocessen på ett sätt som är effektivt, säkert, mer förutsägbart och enklare att kontrollera. Detta får däremot till följd att det framträder ett gap mellan den formaliserade metoden och hur metoden används i praktiken. Den primära utvecklingskontexten är den i vilken systemutvecklingen sker, där utvecklaren står i fokus. I den sekundära, yttre kontexten återfinns hela organisationen. Den innehåller dels oskrivna regler och dels ett företagsklimat som är grunden i organisationens kultur. Att påverka arbetet i en organisation genom att införa en systemutvecklingsmetod kan möta motstånd, vilket i sin tur kan påverka hur systemutvecklingsprocessen utvecklas. (Fitzgerald et al., 2002)

Mot bakgrund av att systemutvecklingsmetoder kan upplevas påträngande i organisationen är det många organisationer som bara använder så pass mycket formaliserad utvecklingsprocess som de anser är absolut nödvändigt. Denna nivå baserar sig på kontextuella och situationsrelaterade faktorer, vilka varierar konstant under hela utvecklingstiden. Coleman (2004) har sammanställt ett antal variabler som påverkar hur mycket formaliserad utvecklingsprocess som anammas av organisationen. (Coleman, 2004)

- Bakgrund för grundare/CIO/ansvarig för mjukvaruutveckling
- Ledarskapsstil och organisationskultur
- Företagsstorlek och storlek på teamet
- Marknadssektor
- Vad som föranleder utvecklingsprocessen

Den minimala processen som används skapas för att tillfredsställda både ledningens krav på resultat och utvecklarnas behov av att vara kreativa. Enligt Coleman (2004) grundar sig en viktig del i problematiken på att personer i ledande positioner i organisationen ser systemutveckling på ett annorlunda sätt i jämförelse med utvecklarna. Ledningen vill gärna sälja in processer för att effektivisera arbetet medan utvecklarna upplever processer som besvärligt, tidskrävande och något som ligger i vägen för det ”verkliga” sättet att producera kod. Detta gör det ofta svårt för organisationen att sälja in processen. En framgångsrik process

kan möta kraven som kommer, både från organisationen samt utvecklarna, där organisationen får en högkvalitativ produkt med rätt funktionalitet i rätt tid till rätt kostnad. (Coleman, 2004)

Organisationer och XP

Organisationer vill ha bevis för att XP fungerar i just deras miljö innan de förespråkar användning av metoden och sedermera även att tillämpa den. Det kan finnas problem med att införa ytterligare en process, särskilt i stora företag som redan har en komplicerad miljö. (Sfetsos et al., 2006). Enligt Sfetsos et al., (2006) uppkommer det olika problem med praktikerna beroende på om utvecklingen sker i en liten eller stor organisation. Även det förfarande med vilket organisationen väljer att angripa problemen skiljer sig åt. Stora företag lägger sig oftare i individuella praktiker som kan inverka på organisationens gängse regler. Ett exempel på detta kan vara ansvarsfördelning som krockar med ”collective code ownership”. Även om praktiken i sig har fått fördelaktiga uttalanden, genom att underlätta för alla inblandade att göra de ändringar som krävs, innebär det samtidigt att en ansvarsfördelning på olika områden inte fungerar. (Fitzgerald et al., 2006) Små organisationer är ofta mer noga med att hitta utvecklare med hög kompetents vilka i sin tur är mer flexibla vid tillämpandet av praktikerna.

Distribuerad utveckling har blivit allt vanligare och är något som kan påverka tillämpningen av XP. Exempelvis kräver parprogrammering fysisk närvaro, medan en praktik som ”collective code ownership” underlättas av enkel kommunikation mellan utvecklarna. Att då få kod levererad “utifrån” som ska integreras, kan innebära bekymmer om kommunikationen inte har fungerat. (Williams et al., 2004)

Det existerar klagomål från utvecklare och ledare att flera praktiker är oprecisa och otydliga. Det är svårt att veta exakt hur exempelvis ”simpel design” och ”metaphor” skall appliceras i en given situation. Trots detta har dessa praktiker ansetts bidra till framgång i utvecklingsarbetet. (Sfetsos et al., 2006) Enskilda praktiker påverkas av organisationen på olika sätt. Praktiken ”customer on-site” vara exempelvis vara svår att tillämpa då kundorganisationen inte alltid vill avvara någon som kan bistå teamet. Istället får någon form av kompromiss tillämpas där kommunikation istället sker via exempelvis telefon eller e-post. I vissa fall är detta den enda lösningen ifall kunden geografiskt befinner sig väldigt långt bort från teamet. (Williams et al., 2004)

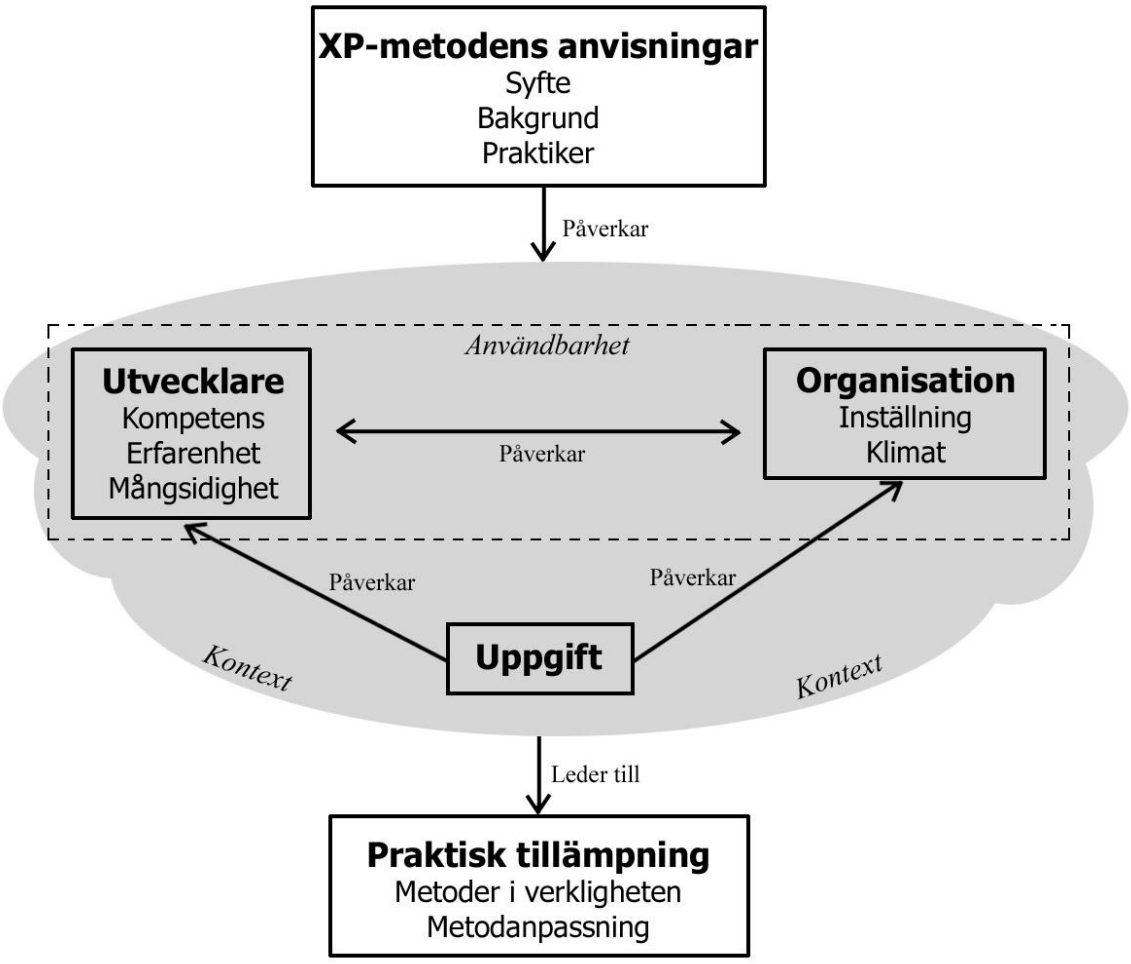
2.4.6 Uppgiften och utvecklingsprojektet

Utvecklingen av informationssystem har tidigare varit väldigt fokuserad kring de tekniska aspekterna av arbetet men Fitzgerald et al., (2002) poängterar att den sociala aspekten är lika viktigt. Alla delar i en utvecklingskontext påverkar varandra. Kontexten behöver hanteras olika beroende på vilken typ av system som utvecklas samt hur resultatet skall se ut. Om exempelvis ett unikt säkerhetskritiskt system utvecklas kräver det en annan typ av kontext än ett generiskt lågrisksystem. Resonemanget tas även upp av Williams et al., (2004) som menar att fokus under utvecklingen bör se olika ut beroende på uppgiften. Exempel på uppgifter som har hårda krav på kontroll är system inom militären och sjukvården medan beslutsstödssystem snarare fokuserar på enkel användning och enkelheten i att förstå systemet. Genom att försöka klassificera den uppgift som projektet står inför kan fokus riktas rätt. Det kan även vara så att uppgiften i sig inte medför frekventa kravändringar eller att releaser görs ofta vilket kan innebära att en agil metod egentligen inte är nödvändig. (Williams et al., 2004)

Fitzgerald et al., (2002) menar att det är viktigt att känna till den mångfald som existerar inom de system som skall utvecklas och hur det i sin tur påverkar utvecklingsprocessen för dem. Försök att kategorisera systemen har tidigare gjorts för att på så sätt skapa en mall för utvecklingen. Till exempel har indelningar gjorts efter storlek, plattform eller användarkategorier. Dock har dessa kategoriseringar inte lyckats fånga den komplexitet som finns inom mjukvaruutveckling. Bättre, mer genomarbetade beskrivningar kan ha ett betydande inflytande över utvecklingsprocessen så som att ange vilken typ av kompetens utvecklarna bör besitta, eller hur uppgiften skall angripas. En skicklig utvecklare skall vara kapabel till att känna av vad det är för system som utvecklas och hur det skulle kunna kategoriseras och därmed kunna hantera systemet utefter det. Exempelvis kan olika system kräva olika typer av utvecklingsprocesser. Ett styrsystem för ett kärnkraftverk behöver till exempel ha en välstrukturerad och kontrollerad process. Som motpol till detta kan nya system som är fullt av radikalt nytänkande passa bättre med en friare, mer kreativ process. Trots detta marknadsför sig flera metoder för att vara just generiska. (Fitzgerald et al., 2002)

2.5 Vårt teoretiska ramverk

Nedan presenteras vårt ramverk som är tänkt att ge en överskådlig bild över hur den insamlade teorin är relaterad.



Figur 2.4 – Vårt teoretiska ramverk

För att ha ett välmotiverat raster att relatera empirin mot, har vi valt att grunda vårt ramverk på tre delar. Dessa delar är Bevan et al:s (1991) definition på användbarhet, Fitzgerald et al:s (2001) ramverk för systemutvecklingsmetoder i praktiken och slutligen Williams et al:s (2004) ramverk för kontexten inom XP-projekt. Användningen av tidigare presenterade ramverk hoppas vi skall ge en större tillförlitlighet för vårt ramverk gällande de komponenter som ingår samt deras relation. Vårt ramverk skall inte ses som en absolut sanning utan snarare fungera som ett raster eller tolkningsverktyg som underlättar analysen av empirin.

Ytterst börjar ramverket med XP-metodens anvisningar. Här inkluderar vi även bakgrunden och syftet med XP. Detta påverkar systemutvecklingskontexten där vi framförallt har fokuserat på komponenterna *utvecklarna* och *organisationen*. Det är hur dessa komponenter upplever *användbarheten* som skall undersökas. Ytterligare en större komponent är den *uppgift* som ska genomföras, varför den presenteras som en delkomponent här eftersom att den pekas ut som en huvudkomponent i användbarhetsdefinitionen samt i båda ramverken. För att tydliggöra att alla delar i kontexten påverkar varandra har ett antal pilar ritats ut, men det är viktigt att notera att kontexten innefattar alla kategorier som nämnts i kontextkapitlet. Resultatet av detta leder till en praktisk tillämpning av metoden.

3 Tillvägagångssätt

I detta kapitel behandlar vi vårt tillvägagångssätt, några metodologiska överväganden samt andra metodteoretiska reflektioner.

3.1 Några metodologiska överväganden

Vi har uppfattat vårt arbete som en problemlösning vilket har medfört att vi arbetat iterativt för att uppnå vårt resultat. Vi gjorde en litteraturstudie som följdes av en förfinad specialisering av forskningsfokus vilken i sin tur har följts av en än mer preciserad litteraturstudie.

För att få en bred infallsvinkel på vår problemställning har vi valt att intervjua ett par personer som har olika kopplingar till XP. Genom att göra på detta vis erhåller vi dels information från ett antal olika synsätt som kompletterar varandra väl, men även mer konkreta gränsdragningar kring vilken inriktning vårt arbete skall ha. Detta på grund av de vaga gränsdragningar vi hade i början av arbetets gång vilka preciserades allt eftersom arbetet fortskred. Detta förfarande gjorde att vi använde oss av kvalitativa studier, där syftet är att ta reda på vilken sort eller karaktär någonting har. Detta gör att vi fångar erfarenheter och innebörden av informantens vardag, vilket är vår mening med intervjuerna och en bidragande faktor för vår precisering av arbetet. (Kvale, 1997).

Kvalitativa respektive kvantitativa undersökningar
Enligt Kvale (1997) finns det två sätt att inhämta kunskap på, genom kvalitativ eller kvantitativ forskning. Kvalitet syftar på sort eller karaktär av någonting och kvantitet på mängd eller storlek. Många studier använder sig av en kombination av båda dessa sätt, dock är den kvalitativa metoden vanligare i slutfasen av en undersökning då resultatet skall rapporteras. Således fungerar dessa två sätt som olika verktyg för att inhämta kunskap och vilket som bör väljas är beroende av de forskningsfrågor som ställs. (Kvale, 1997).

3.2 Vårt tillvägagångssätt

Vårt informationssökande startade genom en litteraturgranskning där vi inledde denna med att läsa Kent Becks bok kring den första upplagan av XP, där Beck presenterar metoden så som han utformat den. Därefter utökade vi fokus till att innefatta litteratur hämtad från diverse bibliotek samt Internet. För att utföra våra sökningar började vi använda oss av sökord av bred karaktär som medgav stor behållning av sökresultatet. Dessa ord var bland annat xp, extreme programming, agil samt agile. Under vår fördjupningsfas där vi ville granska teori på ett djupare plan använde vi oss av mer specifika sökord som tailoring, method tailoring, xp adoption, evaluation study och experience study. Våra sökningar gjorde vi bland annat via Elin@Lund, Google Scholar samt på universitetsbiblioteken. För att inhämta teori som vi

anser aktuell och trovärdig har vi utvärderat den litteratur vi tagit del av utifrån kriterierna nedan.

Källkritiska överväganden
Vid granskning av teori är det viktigt med källkritik. Med källa menas ursprunget till vår kunskap, som kan vara skriftlig, muntlig eller materiell. Källkritik är en samling metodregler som kan brukas för att ta reda på om källorna är sannolika. Gällande trovärdighet, är en grundprincip i källkritik att källan skall vara konfirmerad från två håll för att anses relevant. Detta är dessvärre ingen garanti på att det som bekräftas faktiskt stämmer, då båda parter kan ha fel. Således är detta ett nödvändigt, men inte tillräckligt krav med tanke på efterfrågad trovärdighet. Detta fenomen gäller framför allt när kontroversiella ämnen behandlas, vilket gör att man vid ensamma framställningar kan acceptera att frångå detta handlande och enbart använda sig av huvuddragen, som presenteras längre ner. Thurén (2005). Källkritik hjälper oss att få kontroll på de fakta vi granskar. För att erhålla en relevant faktakontroll finns det fyra källkritiska principer att använda. (Thurén, 2005):

- **Äkthet**. Källan skall vara det den utger sig för att vara.
- **Tidssamband**. Om det har gått lång tid från det att händelsen inträffade till det att den berättas minskar källsäkerheten.
- **Oberoende**. Källan skall vara så fristående som möjligt och inte vara referat från andra källor.
- **Tendensfrihet**. Det skall inte finnas misstankar om att källan medvetet eller omedvetet ger en falsk bild av verkligheten.

Om källkritiken gäller internetkällor är de ovan nämnda principerna fortfarande av vikt, men hänsyn bör även tagas till ytterligare tre kriterier. (Leth & Thurén, 2000).

- Världsbild och kunskapssyn som tendens.
- Trovärdighet.
- Källans förutsättningar och egenskaper.

Att det är fler principer att ta hänsyn till beror på att vem som helst kan lägga ut information på Internet och att det inte finns någon myndighet som kontrollerar eller granskar vad som läggs ut på Internet. Detta medför att informationen kan vara flyktig eller ofullständig. Eftersom att den som granskar information på Internet således är utlämnad åt sitt omdöme är det upp till var och en att ta ställning till den information man möter där. Leth & Thurén (2000). Beträffande var på Internet informationen finns, kan den vara mer eller mindre relevant. De platser där vi har funnit den information vi efterfrågade har varit stora organisationers hemsidor för att öka chanserna för att informationen skall vara fullständig och säker. Vi har däremot enbart använt oss av några bitar på dessa sidor och valt ut de med avseende på tidigare nämnda kriterier.

Allt eftersom vår teoribas har växt har vi även transformerat forskningsfokus och uppsatsens inriktning för att passa gentemot andra författares litteratur kring metoden och dennes situation i dagsläget. Efter den inledande litteratursökningen försökte vi komma i kontakt med något företag eller organisation som jobbade med XP. Detta visade sig problematiskt då vi efter kontakt i form av e-post och telefonsamtal med ett 40-tal företag inte hade funnit någon som hade möjlighet att medverka. Istället besökte vi Lunds tekniska högskolas

arbetsmarknadsdagar, Arkad. Där lyckades vi komma i kontakt med två av våra informanter, via de representanter som stod på mässan. Den tredje informanten kom vi i kontakt med genom en rekommendation från av våra redan etablerade informanter och den sista uppmärksammade vi från några artiklar som informanten skrivit och därefter tog vi personligen kontakt med honom.

Intervjuförfarande
Vi stod inför ett problem gällande mängden informanter. Eftersom att vi hade problem med att få tag i informanter som arbetar med XP fick vi även intervjua de som hade någon annan relation till XP än arbetslivserfarenhet. Därför valde vi att gå mot bredd snarare än djup i uppsatsen. Vi handskades med detta problem genom att i empirikapitlet koppla varje informants syn på XP till delkomponenterna i vårt ramverk för att därigenom få en mer samlad bild över XP. Vår undersökning baseras på fyra intervjuer där tre typer av individer har intervjuats. Genom detta har vi fått information kring praktiskt arbete med XP, en akademisk tolkning av metoden samt en tredje syn på XP från en mer övergripande nivå. Denna blandning av informanter har medfört att vi inte har ställt samma intervjufrågor till alla informanter. Vi ansåg det opassande att exempelvis fråga en person som enbart har akademisk erfarenhet av XP, hur denne har arbetat med XP, vilka praktiker som fungerade bra i arbetslivet och liknande. Detta gjorde att de som har praktisk erfarenhet av XP fick samma frågor, men övriga informanter har specifika frågor. Frågorna återfinns under bilaga 1.

Då antalet informanter inte är så stort har vi därför eftersträvat denna blandning av personer för att på så vis försöka uppnå en genrealiserbarhet. Enligt Kvale (1997) kan en intervju vara strukturerad eller semi-strukturerad. Vi har valt att använda oss av semi-strukturerade intervjuer med öppna frågor som vi anser tillåter informanten att vara mer målande och berättande i sina svar vilket ökar möjligheten för oss att erhålla kunskap som möjligtvis hade fallit bort om vi använt oss av en mer strukturerad intervju.

För att ge intervjun en viss struktur har vi följt Kvales (1997) ramverk som innefattar intervjuundersökningens sju stadier.

- **Tematisering**. Innan intervjuandet börjar skall undersökningens syfte uttryckas samt en beskrivning av ämnet ges. Detta har vi handskats med genom att låta informanten ta del av syftet med uppsatsen samt beskrivit ämnet som vi studerar.
- **Planering**. Planering bör ske för alla de sju stadier som innefattas i undersökningen och även ta hänsyn till den kunskap som eftersträvas och de moraliska påföljder som kan uppstå orsakat av intervjun. Genom att undvika att ställa personliga frågor och enbart beröra relevant information har vi försökt att uppfylla planeringens krav.
- **Intervju**. Intervjuerna ska genomföras enligt intervjuguiden samt med den bakomliggande kunskap som eftersträvas i åtanken. Även den mellanmänskliga relationen skall beaktas. Under intervjun har vi eftersträvat en avslappnad intervju genom att vara sakliga, lugna i framtonen och lyssnat på informanten. Då vi har haft forskningsfrågan i bakhuvudet under hela intervjun har vi indirekt haft möjlighet att styra vissa moment i intervjun i den riktning vi velat.
- **Utskrift**. Materialet skall iordningställas för analys. Detta görs vanligen genom en transkribering där talspråk överförs till skriftspråk. Här har vi transkriberat materialet genom att systematiskt gå igenom den inspelade intervjun och skrivit ner ordagrant

vad informanterna och vi själva sade.

- **Analys**. De analysmetoder som är lämpliga bör väljas med tanke på undersökningens syfte, ämne samt intervjumaterialets karaktär. Då intervjuerna var kvalitativa har vi därför analyserat materialet inte enbart utifrån de intervjufrågor vi ställt, utan även tolkat och kategoriserat det efter delkomponenterna i vårt ramverk.
- **Verifiering**. Intervjuresultatens generaliserbarhet, reliabilitet samt validitet ska fastställas. Dessa faktorer har vi behandlat i detta kapitel, samt under diskussionen i kapitel 6.
- **Rapportering**. Resultatet av undersökningen skall rapporteras med avseende på vetenskapliga kriterier och motsvara dessa. De bör ta hänsyn till de etiska aspekterna av undersökningen och i slutändan resultera i en begriplig produkt.

Efter intervjun med Lars Bendix som undervisar kring XP på LTH fick vi nya idéer kring arbetet, då Bendix fick oss uppmärksamma på de krav som XP ställer på sin omgivning, i huvudsak gällande individen och organisationen. Efter ett möte med vår handledare och en ny litteratursökning var vi inne på det spår som uppsatsen nu har tagit, att behandla användbarheten. Allt intervjumaterial transkriberades och strukturerades varpå vår diskussion där arbetets delar sammankopplades, huvudsakligen vår forskningsfråga, teoretiska ramverk och empirin. Därefter kom vi fram till en slutsats.

3.3 Avslutande metodteoretiska reflektioner

När vi granskar kunskap gör vi det med följande kapitel i åtanke vilken är tänkt att öka objektiviteten, reliabiliteten, replikerbarhet samt validitet.

Bias

Det gäller att granska den kunskap som inhämtas, fri från bias. Syftet är att sträva efter objektivitet för att frambringa kunskap fri från personliga fördomar. (Kvale, 1997) Således bör detta tas i beaktning vid arbete med alla typer av fakta för att inte slutresultatet skall färgas av personen som utför granskningen. Detta har vi eftersträvat genom att utgå från ett öppet sinne och varit öppna för nya tankar och synsätt på teori i allmänhet. Detta är antagligen en svår uppgift, då personliga fördomar är djupt rotade och ibland svåra att inse, men vi har försökt hålla oss neutrala på detta plan.

Reliabilitet och replikerbarhet

Vid forskning inom ett område är målet att undvika att forskningsresultatet dirigeras av den som genomför arbetet. Det som eftersträvas är replikerbarhet; att framtida forskare har möjlighet att erhålla samma resultat om undersökningen eller forskningsarbetet görs om. Att uppnå detta är tämligen problematiskt men kan underlättas genom ett närmande av det som undersöks. Sker detta, kan forskaren påbörja en relation med det han eller hon granskar och därmed försöka förstå informanterna enligt deras egna premisser. (Jacobsen, 2002). Detta har vi eftersträvat genom att dokumentera vårt tillvägagångssätt och lämna primärdata oförändrad i form av intervjutranskriberingar, vilket ökar möjligheten för replikerbarhet.

Reliabilitet är ett begrepp som har spelat en stor roll under hela vår problemlösningsprocess. Kvale (1997) poängterar vikten av att verifiering, som innefattar reliabilitet och validitet, är något som bör uppmärksammas under hela arbetets gång. Reliabilitet behandlar

forskningsresultatets pålitlighet. Genom att hålla en hög reliabilitet så erhålls ett så störningsfritt resultat som är möjligt, fri från påverkan av yttre omständigheter så som ledande frågor i en intervju. Dock kan ledande frågor bidra till ny och värdefull kunskap om de används på rätt sätt. (Kvale, 1997). De ledande frågorna vi har ställt i några av de intervjuer vi har genomfört är planerade, dels för att få en uppfattning om informantens reliabilitet och möjlighet att svara på denna typ av frågor samtidigt som denna typ av frågor i något fall får informanten att förstå våra val av frågor och syftet med dessa.

Validitet
Validitet innebär att det som mäts också är det som var tänkt att mätas. Fenomenet validitet kan även kallas giltighet och för att specificera begreppet ytterligare är det möjligt att dela upp giltighet i två typer, inre respektive yttre. (Kvale, 1997).

Inre validitet
Den interna giltigheten berör frågan gällande om vi har erhållit det vi eftersträvade att inhämta. Det kan finnas många anledningar till om någonting är rätt eller fel, men det som är intressant att studera är huruvida någonting är *riktigt*, om det är *sanningen.* Inom samhällsvetenskapen är *intersubjektivitet* mer intressant än sanningen, eftersom att intersubjektivitet innebär att någonting är så nära sanningen vi kan komma, då flertalet personer är överens om att någonting är riktigt. För att kontrollera den interna giltigheten finns det två möjliga åtgärder, dels en kontroll av den undersökning och de slutsatser som dragits, där dessa jämförs med andra individers arbete, dels en kritisk granskning av själva resultatet. (Kvale, 1997). Vi har gällande den inre validiteten valt att kritiskt granska de slutsatser och resonemang vi dragit för att undersöka om dessa stämmer överens med verkligheten samt sanningshalten kring huruvida det vi erhållit är korrekt.

Yttre validitet
Den yttre giltigheten handlar om i hur stor utsträckning rönen för undersökningen är generaliserbara. Likt den inre giltigheten är det möjligt att införa två typer av generaliserbarhet även på den yttre varianten. Den första varianten innebär en generalisering av data som erhållits från ett mindre urval, som sedan överförs till en teoretisk nivå som innefattar ett större antal enheter än de undersökta. Den andra varianten innebär en generalisering av ett fenomens frekvens, vilket innebär möjligheten att erhålla fakta om en större population från ett mindre urval. Styrkan i den kvalitativa metoden ligger i den första varianten, som innebär att en utveckling mot mer generella teorier, vilket alltså är den vi ämnar undersöka. (Kvale, 1997).

God forskningsetik
Att till varje pris forska kring ett ämne är inte alltid befogat. Ett exempel på detta är forskning som innefattar etiska problem, så som forskning kring atombomben eller manipulation av gener. Idag är det få som tror att forskning är helt fri från påverkan av värderingar. Trots detta är det tämligen få som påstår att all forskning enbart är uttryck för någon politisk, religiös eller dylik tanke. Exempel på sådan forskning är forskning kring ett politiskt parti. Om obehaglig information har dolts så kan forskningen ses som en variant som inte är rättmätig. Detta medför att ingen forskning eller undersökning blir neutral, varpå man istället bör sträva efter *öppenhet*. Detta uppnås genom att vi noggrant beskriver vårt tillvägagångssätt, så att utomstående har möjlighet att få en inblick i vilken information som har, respektive inte har uppkommit vid undersökningen. (Jacobsen, 2002).

4 Presentation av studieobjekten

I detta kapitel presenterar vi informanterna samt respektive organisation som de befinner sig i samt kortfattat beskriver informanternas syn på XP.

De informanter vi har använt oss av kommer från tre olika organisationer. Två av informanterna återfinns på samma organisation. I delkapitlen nedan finns information kring de organisationer som berörs i denna uppsats presenterad, vilken är tänkt att hjälpa i processen av förståelsen för tolkningen av våra empiriska data.

4.1 Softhouse

4.1.1 Allmänt om Softhouse

Softhouse Consulting grundades 1996 och är en leverantör av leanbaserade tjänster inom mjukvaruutveckling. Företaget har tre kontor i Sverige; Stockholm, Malmö samt Karlskrona. Genom andelar i ett annat företag, Qvantel, återfinner man även verksamheter i Finland, Indien samt Singapore. Softhouse är konsultbaserat och specialiserar sig på mjukvaruutveckling av agil typ, i synnerhet används systemutvecklingsmetoden Scrum.

Utbildning i form av bland annat olika typer av certifiering, inom framför allt Scrum tillhandahålls även av Softhouse. Företagets kundbas utgörs av stora samt medelstora företag på olika typer av marknader. Softhouse omsatte år 2007 i moderbolaget Softhouse Nordic AB 96 miljoner SEK.

4.1.2 Informanterna på Softhouse

Två av våra informanter arbetar på Softhouse, nämligen Christian Pendleton och Thomas Lundström. Christian Pendleton har en teknisk bakgrund och har studerat data mellan 1987 och 1994 på LTH. Hans metodintresse startade då han i arbetslivet stötte på metoder i praktiken. Han har coachat ett CM i XP, vilket innebär att han varit mentor åt en som gav stöd åt arbetet och skapade ordning och reda i processen i ett XP-projekt. Christian har arbetat på Softhouse sedan i februari 2008 och är senior konsult. Thomas Lundström har också en teknisk bakgrund med en civilingenjörsexamen i industriell ekonomi. Han håller sitt arbete kodnära och arbetar både med utveckling, projektledning samt testledning.

4.1.3 XP på Softhouse

XP används på Softhouse genom ett val av de delar av metoden som passar in för tillfället. Delar av metoden är dock absoluta krav för projekten, så som kontinuerlig integration och en customer-on-site, enligt Lundström. Metoden följs inte slaviskt eller uttalat då exempelvis parprogrammering enbart används då kunskap skall utbytas mellan utvecklare, vilket ofta görs via en 30 minuters session vid en dator och efter denna återgår de till respektive arbetsstation. Att alla delar bör användas för att nyttan med XP ska uppstå anser inte Lundström, utan han tror mer på ”bottom-up”, att delar plockas in och slutligen inser man att det faktiskt är XP som används. Man bör som programmerare vara duktig för att kunna implementera XP.

XP anpassas med Scrum på Softhouse. XP berättar hur saker ska göras medan Scrum mer befinner sig på en moln-nivå, ett övergripande plan. Denna skillnad anser dock Softhouse är positivt, en styrka för metoderna.

Allmänt så anser Softhouse att XP ger mer mjukvara på kortare tid och med högre kvalitet av färre personer än klassiska metoder. Den ställer även krav på utvecklarna vilket kan ge problem i organisationen. Som utvecklare i ett XP-projekt bör man inte vara helt ny, en god kommunikatör är en förutsättning för att vara en duktig utvecklare. En komplett utvecklare krävs för ett XP-projekt.

Framtiden spår Softhouse till att vara agil. Det är just nu en våg av agila metoder i bruk, men där Lundström tror att många företag kommer att ”bränna sig” på grund av okunskap och omognad till att använda agila metoder överlag. Däremot blir det en spännande utveckling för de som lyckas, vilket kommer ge väldigt mycket på kort tid och med hög kvalitet. Positiv framtid för agila metoder i korthet.

4.2 Create

4.2.1 Allmänt om Create

Create bildades 2004 och är ett företag med fokus på systemutveckling, test/kvalitetssäkring samt teknikinformation. Organisationens ledord visar på att Create vill nå framgång genom att erbjuda kostnadseffektiva lösningar med bibehållen fokus på kvalitet. År 2007 omsatte Know IT Create in Lund AB 186 miljoner SEK. Geografiskt sett är företaget baserat i Skåne och Småland med totalt sex kontor i dessa landskap, men har även ett kontor i Stockholm. Create är verksamt på flera olika marknader, bland annat bank/finans, telekom samt läkemedel, där kundens storlek inte spelar någon roll. Företaget ingår i en större koncern där börsnoterade Know IT är moderbolag. Detta medför att organisationen i stort har drygt 1100 medarbetare med dotterbolag i Sverige, Norge samt Estland.

4.2.2 Informanten på Create

Conny Lundgren som arbetar på Create har en teknisk inriktad eftergymnasial utbildning. Han har arbetat på Create i drygt 6 år och är både konsult och teamchef över javakonsulterna. Kunderna som Lundgren kommer i kontakt med befinner sig både inom den statliga och privata sektorn. Antalet personer som deltar i de projekt Lundgren är delaktig i är allt från några få utvecklare till över hundratalet personer.

4.2.3 XP på Create

XP-användningen på Create är av outtalad art. De använder sig av delar från XP-metoden och blandar dessa i huvudsak med Scrum. Vilka delar de nyttjar beror på rådande situation. Ett faktiskt exempel på detta är i ett pågående projekt hos en kund då plattformen är ny för utvecklarna, vilket medför att de använder parprogrammering för att erhålla en högre hastighet på arbetet.

Synen på XP är att det är en metod som är väldigt utvecklarnära och svår att använda ensam. En blandning mellan exempelvis XP och Scrum fungerar i bra symbios då Scrum sköter hanteringen längre upp i kedjan där management krävs. Detta ser dock inte Create som någon brist utan förespråkar en metodanpassning där de så kallade guldkornen plockas från respektive metod. Däremot kan det finnas tvivel på den extensiva testningen från vissa håll, framför allt innan nyttan med denna visar sig. Överlag är metoden omtyckt av utvecklarna.

Brister i XP-metoden är svårigheten med att implementera metoden i ett redan befintligt projekt som varit igång en längre tid så som ett år eller mer. Då kan det finnas delar i metoden som kräver att de har använts från början men som då saknas i det projekt som skall tas över.

Framtidssynen på agila metoder är god och någonting som Create tror starkt på en agil framtid, om det sedan blir XP som växer eller någon annan metod, det återstår att se.

4.3 LTH

4.3.1 Allmänt om Datavetenskap på LTH

Institutionen för Datavetenskap tillhör Lunds Tekniska Högskola men även den Naturvetenskapliga fakulteten vid Lunds Universitet. Institutionen utbildar bland annat civilingenjörer och håller fristående kurser i ämnet datavetenskap. Forskning bedrivs på institutionen, framför allt inom områdena algoritmer och datastruktur, artificiell intelligens, inbyggda system och programvaruteknik. Det arbetar drygt 70 personer på institutionen, i en blandning av professorer, docenter, disputerande lektorer samt doktorander. Institutionen är placerad på Ole Römers väg 3 i Lund.

4.3.2 Informanten på LTH

Lars Bendix är ursprungligen från Danmark och kom till Lund år 2001. Bendix hade kunskap om XP redan innan han kom till Lund, men det var först i samband med kursen ”Programvaruutveckling i grupp” som han började arbeta med XP. Tillsammans med andra författare har Bendix skrivit några artiklar kring XP. Bendix har ingen egen arbetslivserfarenhet från XP utan besitter istället en akademisk förståelse av metoden.

4.3.3 XP på LTH

Institutionen arbetar med XP genom att erhålla en kurs för studenterna i XP. I denna kurs föreläser Lars Bendix samt agerar kund i de praktiska moment som hålls. Lars Bendix har tillsammans med Görel Hedin och Boris Magnusson skrivit ett antal artiklar kring XP, med

fokus på lärande. Lars Bendix framhäver att XP bör användas fullt ut för att nyttan med metoden skall uppstå, men nämner även att enstaka praktiker kan vara användbara om utvecklarna är erfarna och besitter rätt kunskap. Lars Bendix akademiska erfarenhet kring XP är således stor, men hans erfarenhet kring XP i arbetslivet är mindre.

Lars Bendix poängterar vid ett flertal tillfällen den stora vikten av att ha en kund delaktig i projektet under hela projekttiden. Inte bara anträffbar via tekniska hjälpmedel, utan fysiskt närvarande. Den senaste utgåvan av Kent Becks XP-teori uppskattar Lars Bendix, men av ekonomiska samt strukturella skäl används den första utgåvan i undervisningen.

Möjligheten till anpassning av arbetet och möjligheten att själv strukturera arbetsgången poängterar Bendix som en central punkt i de agila metoderna. Detta anpassas efter varje projekt och dess medlemmar.

5 Resultatet av empirin

I detta kapitel redovisar vi med hjälp av tabeller, citat och löptext vårt empiriska material. De två delkomponenterna i detta kapitel är dels en meningskoncentrering på de svar vi erhållit på våra intervjufrågor från respektive informant, dels ett avsnitt där vårt ramverk ligger till grund för våra empiriska tolkningar. Detta eftersom intervjuerna var av kvalitativ karaktär och således gav mer material än svaren på våra intervjufrågor. Enstaka citat för att belysa några iakttagelser vi gjort förekommer även i detta kapitel.

5.1 Resultatet av intervjufrågorna

Intervjufrågorna till respektive informant presenteras nedan i tabellform, där svaren har koncentrerats för att lättare åskådliggöra resultatet av intervjun. Då inte alla intervjupersoner har fått samma frågor har vi skapat tre olika tabeller som är anpassade efter respektive informants frågor.

Tabell 5.1 Sammanfattning av intervjun med informant 1 (Christian Pendleton)

Frågegrupp	**Fråga**	**Sammanfattning**
Personbeskrivning	*Hur ser din bakgrund ut?*	Har utbildning på LTH inom datavetenskap. Arbetar som konsult på Softhouse och har coachat ett CM i XP.
Typ av XP-projekt	*Vad var det för typ av projekt?*	På ett telekomföretag där ett byte av en processor i en telefonväxelstation skulle äga rum.
	Hur många arbetade i projektet?	Åtta personer.
	Vad var din roll?	Coachade en CM i projektet.
	Hade någon i teamet arbetat med XP tidigare?	Fyra av medlemmarna hade deltagit på Agile Swedens startmöte 2002.
Resultatet av XP	*Vet du hur stor del av XP som användes?*	De körde XP fullt ut, bortsett från "test-first" på grund av tidsaspekten.
	Hur blev resultatet? Var det fördelaktigt att använda XP?	Det var ett försöksprojekt, men blev lyckat. En fungerande lösning togs fram, XP lyckades nå målet.
Positiva och negativa aspekter av XP	*Vad upplevde du som positivt med XP?*	Ökar teamets kommunikation och att man jobbar tillsammans. Skapar medvetenhet hos andra i teamet vad man gör. Mer ansvar på låg nivå i jämförelse med traditionella metoder.
	Vad upplevde du som negativt med XP?	Äkta parprogrammering inte effektivt vid enkel kod. Svårigheter med att slå sig in på hög nivå i företaget. XP skulle inte fungera ensamt, på grund av bristen på projektstyrning.
XP i jämförelse med andra systemutvecklings-metoder	*Hur upplever du XP i förhållande till andra systemutvecklingsmetoder?*	Informationstransparensen är viktigast i agila metoder, är kulturellt betingat. Vattenfallsmetoden är tråkig att arbeta efter, en stor arbetsplan tas fram i början som aldrig stämmer, men man har ryggen fri. Mer ansvar i agila metoder. XP och Scrum fungerar bra tillsammans.

Tabell 5.2 Sammanfattning av intervjun med informant 3 och 4 (Conny Lundgren och Thomas Lundström)

Fråge-grupp	**Fråga**	**Meningskoncentrering C Lundgren**	**Meningskoncentrering T Lundström**
Personbeskrivning	*Hur ser din bakgrund ut?*	Har en teknisk utbildning. Har arbetat på Create i över sex år. Är teamchef över javakonsulterna.	Är civilingenjör i industriell ekonomi. Har arbetat på Softhouse i ett halvår. Har arbetat med projektledning, testledning och en hel del utveckling.
Teoretisk kunskap kring XP	*Hur mycket teori om XP kan du?*	De tolv praktikerna och en hel del bakgrundsfakta.	De tolv praktikerna.
	Känner du till mer än vad praktikerna innebär?	Ja, har läst Kent Becks bok.	Nej, har inte läst speciellt många böcker kring XP.
Praktisk kunskap kring XP	*I vilket sammanhang arbetade du med XP?*	I flera olika projekt. Just nu i ett projekt då en kund ska förnya sin utvecklingsavdelning.	Började med testdriven utveckling och kom in den vägen. Har arbetat med XP i många olika projekt. Just nu i ett litet projekt på fyra personer.
	Hur stor del av XP använde du?	Ett antal, men inte alla praktiker.	Nästan alla praktiker, men inte riktigt alla.
	Vilka delar av XP tycker du fungerar bra att använda?	Parprogrammering, dock inte alltid. Testdriven utveckling. Continous integration.	Parprorgrammering, då kunden inte har exakta specifikationer. Testdriven utveckling, dock inte alltid. Continous integration.
	Vilka delar av XP är svåra att använda?	Parprogrammering kan vara svårt. Aldrig använt customer-on-site.	Parprogrammering. Testdriven utveckling, om man utvecklar något litet.
Positiva och negativa aspekter av XP	*Vad tycker du är positivt med XP?*	Möjligheten att anpassa metoden.	Man får mer mjukvara med högre kvalitet på kortare tid av färre personer, än traditionella metoder. Möjlighet att anpassa XP med någon annan systemutvecklingsmetod.
	Vad tycker du är negativt med XP?	Svårt att implementera på ett redan pågående projekt.	Ställer krav på utvecklarna. Kan ge gnissel i organisationen.
XP:s krav på utvecklare och organisation	*Vilka krav anser du att XP ställer på utvecklaren?*	Utvecklaren bör vara mångsidig, kunna allt. Finns knappt någon sådan i praktiken.	Han bör vara mångsidig, kunna anta många roller. Framför allt testdriven utveckling ställer höga krav på kompetensnivån på utvecklaren.
	Vilka krav anser du att XP ställer på organisationen?	Svårigheten med att sälja in metoden, då den aldrig når högre än projektledaren.	Att en person är ansvarig för kundkontakten. Teamet och organisationen måste vara överens om programspråket.

Tabell 5.3 Sammanfattning av intervjun med informant 2 (Lars Bendix)

Frågegrupp	Fråga	Sammanfattning
Personbeskrivning	*Hur ser din bakgrund ut?*	Arbetar med konfigurationshantering. Har CM-föreläsning på XP-kursen på LTH.
XP:s popularitet	*Är XP populärt nu?*	Industrin vill vara agil, de har hört ett agilt buzz-word.
	När upplevde du att XP var populärt?	År 2001 var det något man pratade om på universitetet men inom industrin var det okänt.
XP:s styrkor och svagheter	*Vilka är XP:s styrkor?*	Fungerar bra då kunden inte vet exakt vad han vill. Omedelbar feedback från kunden. Snabb informationsgång.
	Vilka är XP:s svagheter?	Vill man åt XP:s fulla värde bör man köra hela metoden. Fungerar inte bra om vattenfallstänket är kvar. Svårigheter med att organisationen ska förstå nyttan med en customer-on-site. Utvecklarna ska vara bra på mycket.
Anledning till XP:s position i dagsläget	*Vad är anledningen till att du upplever XP som starkt eller svagt?*	Att Scrum väljs före XP kan bero på certifieringsmöjligheterna samt att XP inte behandlar så mycket projektledning. Kan vara svårt att köra XP fullt ut i praktiken. Ledningens gamla synsätt kan fördärva.
Metodanpassning	*Sker det mycket anpassning av XP?*	XP fungerar bäst i sin helhet. Diskussionen kring metodanpassning är hopplös och bringar inget konstruktivt med sig.
XP i jämförelse med andra systemutvecklingsmetoder	*Hur upplever du XP i förhållande till andra systemutvecklingsmetoder?*	XP är inte svaret på alla frågor. Människorna i teamet är avgörande, att de är crossfunctional. Mer disciplin krävs av utvecklarna vid bruk av XP jämfört med vattenfallsmetoden.

5.2 Empiriskt resultat i relation till vårt ramverk

Eftersom att vi valde att utföra en kvalitativ intervju överstiger vårt insamlade material svaren på intervjufrågorna. För att presentera och använda oss av detta material har vi kategoriserat det utifrån komponenterna i vårt ramverk. Uppdelningen enligt ramverket i detta avsnitt syftar till att ge en mer hanterbar bild av empirin. Nedan följer således en sammanfattning där vi har tolkat empirin från varje informant genom delkomponenterna i vårt ramverk.

XP-metodens anvisningar

***Pendleton*:** Under det första Agile Sweden-mötet år 2002 pratade alla om Kent Beck, att han ansågs vara gud i branschen. Kent Beck säger att dokumentation inte är viktigt, utan att det är det som faktiskt leder fram till en fungerande produkt man skall använda. Man bör inte planera långt i förväg utan jobba med små uppgifter som man arbetar dynamiskt med. Agila metoder kom som en våg och det var svårt att skilja ut vad som var respektive inte var XP, allting agilt var på något sätt en revolutionär tanke. XP är handfast och beskriver hur man ska arbeta. XP-metoden har en väldigt utvecklarnära implementation, där man elakt kan uttrycka det som ett gäng knasiga programmerare i USA som inte gillade dokumentation och projektledare, som därför hittade på en egen metodik. Gällande praktikerna anser Pendleton att testning och kodgranskning är bra, men parprogrammering är meningslöst när man skriver enkel kod. Det är mer eget tänkande och ansvar på låg nivå i agila metoder, till skillnad från traditionella systemutvecklingsmetoder.

***Bendix*:** Kent Beck är väldigt fundamentalistisk. Den andra utgåvan av XP är bättre än den första på grund av att han blivit mer filosofisk och ger en bakgrund varför man använder XP, varför de är bra att göra så samt vilka problem det är man försöker hantera. Dessutom har han delat in praktikerna i två grupper, primary practices och corollary practices. I den andra utgåvan menar Kent Beck även att man kan använda praktikerna i alla kontexter oavsett om man är agil, använder Scrum eller XP. Dock kräver praktikerna att du har de fundamentala praktikerna på plats – de primary practices. Det är möjligt att Beck var för idealistisk i sin första version av metoden.

”Ingen bekant som jag har träffat på säger att de kör allt by-the-book.”
Thomas Lundström, 2008, bilaga B5

***Lundgren*:** XP i sig själv saknar ganska mycket, den är väldigt utvecklingscentrerad. Man behöver någonting som kan hantera kedjan längre upp. Kent Beck var lite av en idealist när han tog fram metoden.

***Lundström*:** Lundström har inte en djup teoretisk bas kring XP utan känner till de tolv praktikerna. Han använder nästan alla praktiker i XP, dock inte på grund av ett aktivt val, utan han tar de praktiker som han tycker fungerar bra och applicerar dessa. Den praktik som används minst är parprogrammering. Lundström har ett antal delar av XP han inte vill vara utan, vilka är configuration management och kontinuerlig integration. Sedan anser han att customer-on-site är bra att ha, men finns det inte så får man försöka handskas med problematiken ändå. Testdriven utveckling är oftast bra, det finns några tillfällen då man kan tänka sig att avstå från det, exempelvis när man utvecklar något väldigt litet eller oviktigt. Att man anpassar metoden i praktiken är ingen svaghet, utan snarare en styrka.

Utvecklare

***Pendleton*:** Det team som utförde uppgiften tror Pendleton hade klarat av det oberoende av utvecklingsmetodik, då de redan hade arbetat ihop väldigt mycket tidigare i andra projekt. Han anser även att om man har duktiga människor i projektet spelar det ingen roll vilken metodik de använder sig av, för det kommer alltid att bli bättre i jämförelse mot någon som inte har lika mycket erfarenhet, det är viktigare att ha bra människor än att ha en bra metodik. Duktiga människor löser alltid problemen. Det är extremt vanligt, särskilt på stora företag, att många utvecklare gömmer sig bakom en process, de har jobbat på ett visst sätt i tio år och det är tryggt och bra anser de. Pendleton ställer sig frågan om dessa fungerar i en agil värld. På Softhouse är det många kreativa människor, framför allt på verksamhetsutvecklingsavdelningen där Pendleton sitter, man pratar om att folk här som krokodiler – stor mun och inga öron, vilket stämmer.

”Det är viktigare att ha bra människor än att ha en bra metodik. Duktiga människor löser alltid problemen”.
Christian Pendleton, 2008, bilaga B2

***Bendix*:** Om man är en erfaren programmerare och har en projektledare med erfarenhet så vet man vilka saker som fungerar respektive inte fungerar. Om man inte har så stor koll, eller om projektet går trögt så har man hjälp och stöd från XP, som då ger en anvisning om hur man bör göra. XP ställer stora krav på programmeraren genom att denna skall vara duktig på flertalet delar, så som att skriva testfall, programmera och testa vilket medför att alla programmerare klarar inte av att arbeta med XP, men om utvecklaren är erfaren kan det till viss del kompensera behovet av alla praktiker. Beroende på vilken projekttyp och vilken typ

av utvecklare det är i projektet kan man anpassa metoden, till exempel om det behövs skrivas mycket dokumentation så gör man det. Genom att ha mer kunskap kring vad som fungerar kan man nå djupare i det agila tänkandet och har möjlighet att tillämpa metoden som man själv vill. Om Bendix har duktiga medarbetare i sitt team och dessa är crossfunctional, som det kallas i Scrum, skulle han inte ha något emot att använda XP eller Scrum som utvecklingsmetodik.

***Lundgren*:** XP ställer stora krav på teamet i sin helhet. Man kommer aldrig lyckas implementera XP i ett team med väldigt juniora utvecklare, samtidigt som han tror att man får svårt att göra detsamma i ett team bestående av väldigt seniora utvecklare, då de ofta har en uppfattning om hur man alltid har gjort, så kallat tröga i förändring. En bra mix av juniora och seniora utvecklare är bra, vilket också passar in med hur verkligheten ser ut. En utvecklare i ett XP projekt måste kunna anta många roller, vilket är en svår uppgift i praktiken, det är svårt att hitta folk som kan allting. Ofta är de specialiserade på någon del, men det som Lundgren anser att alla skall ha koll på är testdelen. XP brukar vara väldigt överväldigande för utvecklarna att börja med. Framför allt testningen brukar ifrågasättas i början, att det tar lika lång tid att skriva tester som själva koden, men de ser nyttan med det efter första missödet på grund av uteblivet test. Metoden är ganska omtyckt av de flesta utvecklare, bortsett från de som tycker att ”så här har vi aldrig gjort förr”. I parprogrammeringen behöver det inte vara just två utvecklare, utan Lundgren provade i ett projekt att låta testare samt utvecklare parprogrammera tillsammans. De hade helt olika perspektiv på test vilket bidrog till ett lyckat resultat. Synen av att utvecklaren skall kunna allt tycker Lundgren är jättebra på pappret, men det finns knappt några sådana personer i verkligheten.

***Lundström*:** Att använda sig av testdriven utveckling ställer stora krav på utvecklaren anser Lundström. Man bör vara en mycket duktig mjukvarudesigner, man bör kunna se tre, fyra steg i förväg vad som kommer hända. Man måste vara väldigt komplett som utvecklare för att kunna nyttja XP.

Organisation

***Pendleton*:** Telefonväxelverksamheten är traditionellt stel i sitt metodarbete vilket ställer metodarbetet i en lite speciell situation säger Pendleton. Det som är viktigast i agila metoder men som företag har störst problem med är informationstransparensen – alla skall ha tillgång till all information. Det är en kulturell fråga och ett tankesätt som är svårt att få igenom. XP som är en utvecklarnära metodik befinner sig enbart på teamnivå, men exempelvis Scrum som riktar sig till projektledaren som har rätt så mycket formell och informell makt, vilket gör det enklare att få igenom metoden på ledningsgruppmöten och liknande.

***Bendix*:** Om man kör Scrum så borde den som är Scrummaster se till att teamet fungerar på ett bra sätt och arbetar smidigt genom att titta mer på XP. Många organisationer uppfattar XP som en metod med anarki, ”man kan göra som man vill, man behöver inte metodik”. Detta är helt fel anser Bendix– man behöver mer disciplin i agila projekt än i vattenfallsprojekt. Organisationen vill försäkra sig om att ha koll på projektet, vilket får dem att välja någon annan metodik med föregående resonemang i tankarna. I vattenfallsprojekt kan man ta fram sin handbok och läsa sig fram till nästa steg, men i Scrum och XP finns det bara riktlinjer, som säger ungefär hur man ska göra. Är det fortfarande vattenfallstänk i organisationen trots att de använder XP fungerar inte metoden menar Bendix.

***Lundgren*:** Med avseende på vilken metod kundens organisation tidigare har arbetat efter går Lundgren efter erfarenheten och ser vilka förutsättningar som ligger till grund och väljer det

som han anser passar, exempelvis om parprogrammering skall användas om medlemmarna i utvecklingsteamet ligger på olika nivå. De flesta implementationer Lundgren har sett innefattar både Scrum och XP. Det är även så att metoden kommer ”bottom-up”, aldrig ovanifrån. Organisationens anledning att införa XP skiljer sig ganska ofta mycket från utvecklingsteamets perspektiv. Gällande praktikerna är parprogrammering är den svåraste biten att sälja in i organisationen. Det är svårt att förklara varför vi ska ha två utvecklare som gör någonting när vi kan ta två utvecklare och utföra dubbelt så mycket arbete. Det är däremot mycket lättare att få in det som en inlärningsbit när man ska ta in nya personer i projektet eller öka kompetensen på de som redan finns. Att det saknas certifiering i XP kan påverka organisationen till viss del. Just Scrumcertifieringen som jag har, är i praktiken noll värd. Man kan sova sig igenom kursen, det är något man får på papper enbart. Försöker man sälja in metoden högre upp i kedjan så kanske certifieringen är bra. Men jag är inne på att de fyller olika luckor, Scrum och XP. Kör man Scrum är det vanligt att du tar systemarkitekten eller projektledaren som får bli Scrummaster. Genom detta får du in mer businesstänk i arbetet, uppåt.

***Lundström*:** XP ställer krav på organisationen genom att eftersträva en customer-on-site. Finns inte denna på plats så blir kommunikationsvägarna mycket långsamma. En annan faktor för organisationen att ta ställning till är den att XP inte säger vilket programspråk man ska använda sig av, utan det gäller att teamet och organisationen är överens på denna punkt.

Uppgift

***Bendix*:** XP fungerar inte på en projekttyp där XP inte är lämpligt, då uppgiften spelar roll vid arbetet. Ett exempel på detta är vid XP-användningen i utbildningen då studenterna tar fram en layout där siffrorna som skall visas skall vara lättavlästa. Om kunden inte har varit på plats och blivit tillfrågad gör de oftast siffrorna för små, och har ibland stora problem med att ändra layouten så att den passar tänkt användningsområde.

***Lundgren*:** I det projekt Lundgren arbetar med just nu behöver kunden förnya sin utvecklingsavdelning. De kommer byta teknik och kommer att titta på hur de arbetar rent praktiskt. I den situationen använder de sig av en agil metod med små iterationer och parprogrammering, eftersom att en del av utvecklarna är ganska nya på plattformen. För att de skall komma upp i hastighet har de valt att dela upp det så. XP fungerar nog betydligt bättre om man börjar från början i ett projekt, att implementera det i efterhand är betydligt svårare. Har man haft ett projekt rullande i 1-2 år så är det ganska svårt att implementera XP i det, de kanske saknar vissa viktiga delar. En statlig myndighet skulle byta teknisk plattform, vi hade 18 utvecklare uppdelade i två Scrumteam. Alla utvecklare var seniora, men på grund av att plattformen var helt ny så använde vi parprogrammering för att träna folk, där parprogrammeringen senare fasades ut. Det var ett viktigt projekt som gick bra.

”Det är inget självändamål att följa metoden.”
Conny Lundgren, 2008, bilaga B4

Användbarhet

***Pendleton*:** Gillar att man med agila metoder ökar teamets kommunikation och att man jobbar tillsammans, vilket främjar bra lösningar och ett kollektivt kreativt tänkande samt en medvetenhet om vad andra i teamet gör. Att XP inte fungerar skulle kunna bero på att metoden inte kommer upp på dagordningen på beslutande möten där man i praktiken bestämmer, där pengarna och makten finns. XP kan inte fungera ensam i en stor organisation

då den inte adresserar tillräckligt mycket projektstyrning. Vissa av praktikerna i XP skulle man kunna ta och tillämpa i andra systemutvecklingsmetoder.

***Bendix*:** Att anpassa metoder är en hopplös diskussion som inte tillför något konstruktivt. Styrkan i XP är den snabba feedback som ges samt den snabba gången i projektet, eftersom att vi inte ska koordinera och kommunicera så mycket samt vid kommunikation - de direkta kommunikationsvägarna.

***Lundgren*:** Lundgren menar att så fort metoden är i vägen för det dagliga arbetet tappar den sitt värde. Han pekar också på att det inte är något självändamål att följa metoden.

***Lundström*:** Lundström anser att man får mer mjukvara med högre kvalitet av färre personer på kortare tid när man använder XP istället för någon klassisk systemutvecklingsmetod. Man blir produktiv om man använder de praktiker som finns i XP.

Kontext

***Pendleton*:** Olika beroende på den traditionella kulturen, exempelvis Sverige kontra Danmark, där Sverige har lättare för att acceptera det agila tankesättet.

***Bendix*:** XP fungerar bra i vissa kontexter. Speciellt i den kontext då kunden inte har full koll på vad han eller hon efterfrågar, är beredd att vara customer-on-site och är beredd att vi tar fram det här med hjälp av iterationer. Om kunden har koll på allt från början och inte är beredd att ha en customer-on-site fungerar vattenfallsmetoden lika bra. I kontexter som kräver omfattande dokumentation fungerar XP mindre bra, då metoden innehåller mindre byråkrati och litet fokus på dokumentation.

***Lundgren*:** Utifrån varje projekt plockar man de praktiker man behöver använda.

Praktisk tillämpning

***Pendleton*:** Projektet som Pendleton coachade var väldigt seriösa vid användandet av XP och försökte få med alla grundparadigmer, vilket också lyckades, de körde XP fullt ut. Alla praktiker utom ”test-first” användes då denna i praktiken var oanvändbar på grund av tidsaspekten – att skriva test till allt som redan var utvecklat skulle ta många år. Projektet blev mycket lyckat, en fungerande lösning togs fram med hjälp av XP. XP och Scrum fungerar bra tillsammans, projektstyrningen från Scrum och den tekniska biten från XP spelar bra tillsammans. Skall man hårddra det så är Scrum en projektstyrningsmetodik och XP är en utvecklingsmetodik, med ett visst överlapp, menar Pendleton.

”Om du verkligen vill ha det fulla värdet av XP så ska du köra det helt, självklart.”
Lars Bendix, 2008, bilaga B3

***Bendix*:** 2001 var XP något man pratade om på universitetet men inom industrin var det okänt. Om man vill ha det fulla värdet av XP bör man köra det fullt ut hävdar Bendix. Men om ett företag säger att man vill använda sig av testdriven utveckling, varför skulle de inte kunna göra det i så fall? Däremot om de vill använda sig av någon praktik från corollary practices måste man se vad som krävs i primary practices för man säkert skall kunna använda sig av den tänka praktiken. Man behöver erfarenhet för att värdesätta metoden. Den första utgåvan med sina tolv praktiker är mycket kort och precis, lagom för de studerande att ta in. Man behöver konkret användning av XP för att förstå nyttan med den andra utgåvan. Att

Scrum används mest i arbetslivet idag tror Bendix beror på de certifieringar man kan erhålla inom metoden. Att kunden är på plats är mycket viktigt, på grund av att man inte själv skall ta beslut för kundens räkning. Det som också har upptäckts är att man i vissa fall inte har behov av en partner, utan at man istället använder kodgranskning. Detta sker då man har enkla uppgifter att sköta. Inom industrin ser man det som att två personer gör en persons arbete, men det är fel. Det är två personer som gör två personers arbete. Att ha nära till customer-on-site är mycket viktigt. Det räcker att avståndet är mer än 25-30 meter för at meningen med customer-on-site försvinner – då kan de lika gärna sitta på månen. Även om alla inte gör allt i projektet kan man köra XP ifall det finns tillräckligt med expertis i teamet så att alla roller fylls. Bendix uppskattar XP väldigt mycket men anser att inte är svaret på alla frågor.

***Lundgren*:** Har inte använt alla tolv praktiker, men gillar parprogrammering, däremot inte alltid, stand-up-meeting och testdriven utveckling är bra. Använder även continous integration. Man bör inte vara helt renlärd efter det som skrivs, utan man plockar det som känns vettigt, oavsett vilken systemutvecklingsmetod man använder. Customer-on-site har Lundgren aldrig använt sig av och har ingen åsikt kring. Lundgren har aldrig kört XP för sig själv, utan anpassar metoden med exempelvis Scrum, då han anser att metoden fungerar bra men inte är komplett. Konflikten i metoden ligger i hur man implementerar vissa saker. Projektledaren och beställaren har ofta olika perspektiv på hur projektet skall göras, det finns ett glapp mellan XP och verkligheten där. Däremot är detta ingen brist i XP, utan genom att anpassa metoder och plocka delar från olika metodiker så kan man bygga en egen metodik och på så sätt få hela kedjan. Lundgren tror att det är ett säkert sätt att misslyckas med XP då man försöker införa allt, framför allt på all på en och samma gång. När man inför allt samtidigt från man väldigt mycket overhead på varje sak man skall göra, vilket leder till motstånd. Så fort metoden kommer i vägen för det dagliga arbetet så tappar den sitt värde, i slutändan är det du som utvecklare som skall leverera kod. Det är inget självändamål att följa metoden. Agila metoder anser Lundgren vara rätt väg att gå, sedan återstår det att se om det är XP som gäller i framtiden eller någon annan agil metod. Man måste köra XP ett tag innan man kan säga om det fungerar eller inte, och det är dit vi har kommit idag, vi har sett att det fungerar. Scrum och XP är inte konkurrenter, utan kommer snarare gå ihop. De är ett ganska stort överlapp idag mellan dessa två metoder. Scrum hanterar projekt i större skala. Scrum säger hur projektet ska köras och XP är för det praktiska arbetet. Scrum och XP kombineras omedvetet, det är som en verktygslåda – man tar de verktyg man behöver. Vad Lundgren har sett når XP aldrig högre än till projektledaren. XP är ofta något man gör och inget man behöver sälja in menar Lundgren. Ingen av praktikerna i XP är nya, utan XP är mer ett samlingsnamn för ett antal praktiker, där man sedan tar vad man behöver så att XP mer fyller en upplysande roll än att vara ett regelverk. Kent Beck vill att man ska köra hela metoden, men det fungerar inte. Välj de praktiker som fyller en funktion.

”...så har jag inte kört XP rent, det är en bra match med andra metoder...”
Conny Lundgren, 2008, bilaga B4

***Lundström*:** I det projekt Lundström sitter i för tillfället använder de sig av customer-on-site, dock finns de inte i samma rum. Produktägaren är däremot i samma rum som utvecklarna. Man blir lite väl improduktiv om man fastnar i testdriven utveckling. TDD-gänget argumenterar för att TDD bygger upp en arkitektur och design, vilket Lundström inte håller med om. Han anser att man bör ha en grunddesign utstakad när man börjar. Man måste tänka lite ”up-front”. Parprogrammering fungerar bra i vissa fall, framför allt när man ska utarbeta något när man saknar specifikationer. Alltså, när man ska sprida kunskap eller arbeta fram en design är parprogrammering bra. Man tar det när det behövs, i korta intervall. Han anser att man inte måste följa XP slaviskt för att det skall fungera. Lundströms har aldrig försökt att

implementera allt i XP och inte sett någon annan göra det heller. Han tror snarare att man utgår från andra sidan, att man använder ett par praktiker och sedan upptäcker man att man använder allt som finns i XP. Man gör det ”bottom-up”. XP och Scrum kompletterar varandra väldigt bra. Scrum förklarar att man jobbar i iterationer, mer på molnnivå medan XP säger hur man praktiskt jobbar i dessa iterationer säger Lundström.

6 Analys & diskussion

I detta avsnitt analyseras och diskuteras empirin samt sätts i relation till teorin. Grunden för relationerna diskuteras utifrån vårt ramverk. I detta kapitel förekommer ingen ny teori eller empiri. De upprepningar som görs sker enbart i syfte av att förtydliga vårt resonemang.

Det är inte svårt att förstå Becks bakomliggande syfte med att skapa en ny utvecklingsmetod. Under projektet på Chrysler där han systematiskt samlade in praktiker som han ansåg fungera väl ger han ett intryck att ha som mål att skapa den ultimata, heltäckande processen. Det stora antalet praktiker samt det faktum att alla måste användas ger ett intryck av att det lätt skapas en hög arbetsbelastning. Detta styrks av att både Bendix och Lundgren kallar Beck för idealist och Bendix går så långt att kalla honom för fundamentalist.

Att Kent Beck har en teknisk bakgrund med en mastersexamen i datavetenskap anser vi genomsyrar metoden. Då Beck försökte lösa de problem han såg i arbetslivet som systemutvecklare är det inte svårt att förstå att metoden baseras på utvecklarnära kunskap. Vi får medhåll i detta påstående från både Lundgren och Pendleton som menar att metoden är utvecklarcentrerad och både Lundgren och Lundström uttrycker att metoden inte i tillräcklig utsträckning stödjer projektstyrning. Pendleton nämner att man elakt kan uttrycka det som att metoden är ett resultat av en grupp programmerare i USA som inte gillade vare sig dokumentation eller att ha en projektledare. Därför arbetade fram de fram en egen systemutvecklingsmetod som Bendix kallar för anarki för systemutvecklarna. Att XP-metoden relativt sätt innehåller få praktiker som uttryckligen ska stödja projektstyrning samt att flertalet praktiker ger utvecklarna ökad makt stärker denna hypotes. Men att en utvecklarfokuserad metod automatiskt innebär ett lyckat resultat är inte självklart och det innebär inte heller att utvecklarna tycker att den är komplett. Till exempel anser Lundström att metoden inte har alla delar som krävs för mjukvaruutveckling, men att detta inte är en svaghet hos metoden, utan snarare en styrka. Anledningen är att praktiker kan väljas utefter det avsedda behovet. Detta strider helt mot Becks holistiska tankesätt. Trots detta har informanterna Lundgren och Lundström erfarenhet av att använda metoden på ett anpassat sätt med lyckat resultat. En intressant notering är att Bendix är den av informanterna som är mest positiv till att använda XP i sin helhet, då han hävdar att XP då fungerar som bäst.

Konceptet av de tolv praktikerna som omfattas av XP kan vara enkelt att förstå men har ibland visat sig vara svårt att använda i praktiken. Pendelton menar att metoden är handgriplig att arbeta efter vilket stärker Becks syfte att hjälpa en systemutvecklare i arbetet. Däremot får Beck kritik av Pendleton gällande praktikerna – vissa är meningslösa hävdar han. Parprogrammering tar han som ett exempel. När man skriver enkel kod anser Pendleton att denna praktik inte fyller något syfte utan kan snarare ses som tidsödande. Att Lundström och Lundgren använder XP, men inte fullt ut, kan tyda på att metoden är inte är helt anpassad efter de förutsättningar som finns.

Beck har antagligen insett metodens begränsningar och därför utarbetat en andra utgåva av metoden. Andra utgåvan, säger Bendix, ger bakgrunden till metoden ut ett mer filosofiskt perspektiv. Den är även mer strukturerad efter praktiker som är uppdelade i grundläggande

respektive utökade. Beck vill ha de grundläggande på plats i ett projekt innan de andra kan börja användas.

Utvecklaren

Fitzgerald et al (2002) nämner att det är viktigt att poängtera att det är utvecklarna och inte metoderna i sig som utvecklar systemen. Detta är något som styrks av flera av våra informanter. Pendleton går till och med så långt som att hävda att duktiga programmerare kan leverera ett bra resultat oavsett vilken metodik som används, eftersom de innehar kunskap om hur många olika problem ska lösas. Bendix menar istället att erfarenheten ska användas till att ha någon form av distans till utvecklingsmetoden och att denna bör användas till att avgöra vad som fungera respektive inte fungerar med en metod. Erfarenhet gällande systemutveckling anses alltså kunna kompensera för brister i en metod.

Pendleton menar att människor är bättre på systemutveckling än metoder eftersom de är så kreativa. Att då Fitzgerald et al (2002) deklarerar att systemutvecklingsmetoder hindrar kreativitet skulle styrka denna hypotes. Bakgrunden ligger att de menar på att eftersom att systemutveckling inte sker på ett metodiskt sätt, så behövs kreativitet för att projekten ska fungera. Att då ha erfarenhet och kreativitet som kan täcka för brister eller svagheter i en metod skulle då rimligen kunna ha betydelse för processen och i slutändan resultatet. Någon som går emot detta är Leonard-Barton (1987) med sitt resonemang gällande att programmerare med erfarenhet tydligare ser nyttan med att använda sig av metoder och därmed är mer benägna att använda dem. En möjlig förklaringsmodell kan skapas genom att koppla till Fitzgerald et al:s resonemang om att systemutvecklingsmetoder i början fungerar som ett stöd. Därefter anses de hindra arbetet för att slutligen användas på nytt men då på ett skräddarsytt sätt. Skulle detta stämma innebär det att de personer som Pendleton varit i kontakt med skulle befinna sig på mittennivån erfarenhetsmässigt sätt, och rimligheten i det kan diskuteras. I sammanhanget ska dock noteras att Leonard-Barton presenterade sina resultat år 1987, drygt tio år innan agila systemutvecklingsmetoder blev populära.

Fitzgerald et al:s utökade förklaringsmodell skulle då rimligen kunna vara mer anpassad efter nuvarande metoder och forskning vilket följaktligen leder till en högre validitet. Intressant i sammanhanget är Lundgrens påstående om svårigheten med att införa agila systemutvecklingsmetoder hos väldigt seniora utvecklare. Anledningen är att de redan har klart för sig hur utveckling bör ske och systemutvecklingsmetoderna har inte ansetts vara tillräckligt attraktiva i deras ögon Gällande juniora utvecklare menar Lundgren att teamet som helhet då inte skulle fungera om det endast bestod av dessa. Lawrence (2007) visar upp ett sådant experiment och pekar just på flertalet större brister som uppstår vid ett sådant upplägg. Bristen på erfarenhet skapar problem när koden utvecklas och den växer oftare fram på ett felaktigt sätt. Han menar att metoden i sig inte räcker för att styra arbetet. Pendleton är inne på samma linje och ser metoden som en bra vägg att luta sig mot, något som ger vägledning.. Det finns således ingen entydig bild av vilken typ av systemutvecklare, ur ett erfarenhetsperspektiv, som är mest öppen för att använda, uppskatta eller vara hjälpt utav XP.

Resonemanget kan fördjupas ytterligare då flera informanter är överens om att XP ställer stora krav på systemutvecklarna. Bendix menar att det krävs goda kunskaper inom flera olika discipliner för att behärska XP och Lundgren menar att just det stora antalet olika roller som utvecklarna måste besitta är besvärligt. I praktiken är det helt enkelt svårt att hitta en sådan djup kompetensbredd hos utvecklare. Lundgren nämner exempelvis att de utvecklare som är specialiserade i test mycket väl kan bidra till att bättre tester utformas, men problemet är att alla utvecklare inte är specialiserade inom test finns med andra ord redan här utrymme för försvagningar av metoden. Lundström tar dessutom upp test som en mycket svår disciplin att

bemästra. Test framstår med andra ord som en omfattande disciplin som ställer stora krav på utvecklaren, ett område som i sig är stort nog att behärska ensamt. När då stressen innefinner sig i ett projekt ligger det nära att anta att de delar som upplevs som svåra delarna, så som test, är de som först faller bort och det är precis det som Karlström (2002) upplevde i sin forskning. Att som Beck hävda att samtliga praktiker alltid ska följas, eftersom det är då man kan arbeta som mest effektivt, förefaller vara ett bräckligt resonemang som inte är väl förankrat i praktiken. Dock har det visat sig fördelar med att kräva denna bredd hos utvecklarna och det är att metoden skapar en bryggande funktion där kunskap lätt sprids och utvecklare som är specialiserade på något område får en större bredd. Lundgren hävdar just detta och insikten av de olika delarna i utvecklingsarbetet blir tydligt. Detta kräver dock, precis som Murru, Deias & Mugheddu (2003) uttrycker, att alla i teamet får ta plats och därmed få ta del in i denna process. XP skulle med andra ord kunna ses som en lärande miljö för systemutveckling något som visats sig fungera väl av Hedin, Bendix och Magnusson (2003), då XP-metoden användes med lyckat resultat på en kurs i programvaruutveckling i grupp.

Organisationen

En systemutvecklingsmetod ska enligt Fitzgerald et al. (2002) hjälpa till att hantera den komplexitet som systemutveckling innebär. En tillämpning av XP behöver därför vissa organisatoriska förutsättningar för att fungera som det är tänkt. Pendleton nämner en sådan till synes trivial sak som att alla i teamet ska ha tillgång till all information samt att alla ska kunna göra ändringar var som helst i koden. Det är inte en självklarhet att alla företagskulturer tillåter ett sådant förfarande. Detta ligger i linje med vad Coleman (2004) och Fitzgerald et al (2002) nämner att just organisationskultur är en viktig faktor för hur mycket systemutvecklingsmetoder används i en organisation. Vidare så kan det få konsekvenser på en mer konkret nivå. Att till exempel sälja in parprogrammering eller customer-on-site nämns av våra informanter som något problematiskt. Ledningen i organisationen kan ha svårt att förstå hur två personer producerar mer tillsammans när de arbetar med bara en dator.

Vidare är customer-on-site något som Bendix menar på är svårt för organisationen att ta till sig. Att frånvara en resurs till ett utvecklingsprojekt kan ses som slöseri med dennes tid, men Bendix att det är viktigt att ha kunden nära in på sig för att kunna ställa frågor. Annars är risken stor att utvecklarna antar saker eller försöker lösa problemet ändå utan att stämma av med kunden. Här noteras en uppenbar risk. Den bristande förståelsen för metoden hos organisationen verkar skapa problem. Trots att det finns fördelar med att ha nära tillgång till kunden och att det mycket väl kan skapa mervärde att ha med kunden på plats i projektet så är det svårt att sälja in. Ett möjligt antagande för att underlätta detta är att en organisation eller en ledning som har en förståelse för programmering och vad det innebär. Möjligen skulle de då lättare skulle kunna acceptera metodens krav. Detta är också något som bekräftas av både Bendix och Pendleton. XP är en metod skapad av utvecklare och det är ofta från utvecklare som förslaget kommer att använda den. Pendleton nämner just att införandet av XP aldrig kommer ovanifrån i organisationen utan att det är utvecklarna själva som önskar det. Det kan tänkas att ett mindre företag, till skillnad från många stora företag, har färre hierarkiska nivåer, vilket gör att ett initiativ längst ner i organisationen lättare får genomslag. Något som bekräftas av Coleman (2004).

Denna svårighet som XP har att sälja in sig hos organstationen kan vara en möjlig anledning till att flera informanter nämner systemutvecklingsmetoden Scrum och belyser dess projektledningsfunktioner. De menar att Scrum är lättare att sälja in hos ledningen då den är enklare för dem förstå och därmed ser de tydligare nyttan med den. En möjlig förklaring till

denna brist hos XP kan ligga i metodens bakgrund. Att metoden är skapad av utvecklare för utvecklare kan mycket väl ligga metoden till last för att nå ett stort genomslag. Fördelarna kan kanske verka tydliga för utvecklarna, men för organisationen är det inte lika tydligt och konkret varför metoden skulle attraktiv. Ur ett användbarhetsperspektiv är denna iakttagelse speciellt intressant. Coleman (2004) nämner att organisationer vill använda en så minimal metod som möjligt och XP kan då mycket väl framstå som för komplicerat. Dessutom hävdar Stefsos et al (2006) att det är viktigt för organisationen att förstå att XP passar in i just deras miljö och därmed är till nytta just för *dem*. XP har alltså inte lyckats fullt ut med att visa sig attraktiv ur organisatorisk synvinkel och uppvisar även en bristande insäljningsförmåga. Två saker som är nära kopplade till varandra.

Uppgiften

Fitzgerald et al (2002) nämner att det tidigare var de tekniska aspekterna av systemutvecklingsarbetet som hade ett dominerande fokus i projekten och att den sociala aspekten åsidosattes. Bendix nämner ett exempel på detta där en uppgift i ett projekt var att definiera en lagom stor display till ett system. Om utvecklarna då hade tagit ett steg bakåt och funderat på vad som egentligen var syftet med displayen hade de antagligen genast upptäckt bristen i systemet. Här hade ett tillämpande av praktiken customer-on-site, där kunden hade rådfrågats, genast givit svar på frågan. Genom att balansera utvecklarnas tekniska kunskaper med en kunds praktiska tänkande kan uppgiften som ska genomföras nå ett för kunden, mer önskvärt resultat. Det kan tänkas att denna balans varierar med uppgiftens karaktär om hur ofta kunden får verklighetsförankra utvecklaren. Här ser vi ett exempel på hur XP kan öka sannolikheten för att kunden ska få ett önskvärda resultat.

Även seniora utvecklare kan uppleva utvecklingen som svår beroende på uppgiften som ska utföras, påpekar Lundgren. Fitzgerald et al (2002) tar upp flera försök som gjorts med att kategorisera system för att kunna skapa en mall för utveckling. Samtidigt menar de att en skicklig utvecklare skall kunna känna av vad det är för system som ska utvecklas och anpassa processen efter det. Det skedde precis så, enligt Lundgren, då parprogrammeringen tillämpades i ett projekt med en besvärlig uppgift. Trots att alla 18 deltagare var seniora gav en praktik dem fördelar och effektiviserade arbetet. Intrycket från detta är att seniora utvecklare mycket väl skulle kunna vara bättre på att anpassa sig till olika uppgifter än junioira, mindre erfarna utvecklare, dock finns det en begränsning för hur effektivt arbetet kan bli utan en guidande process.

Praktisk tillämpning

Gällande den praktiska tillämpningen av XP anar vi en skillnad mellan den akademiska världen och näringslivet. Fitzgerald et al (2006) tog som bekant upp att de flesta framgångshistorier som redogjorts var inom den akademiska världen och Beck har som vi tidigare nämnt en akademisk bakgrund. Att då Lars Bendix, som arbetar på institutionen för datateknik, anser att om någon ska ha nytta av XP skall de implementera hela metoden är en intressant notering. Detta synsätt strider mot de informanter som har använt XP i praktiken. Lundgren anser att XP fungerar bra men att det inte är en komplett metod. Vidare nämner han att han själv aldrig har kört XP fullt ut, utan blandar med någon annan systemutvecklingsmetod, som exempelvis Scrum. Lundström har på samma sätt aldrig försökt implementera hela XP, eller ens sett någon göra det. Vi tolkar detta som att metoden vid praktisk användning inte fyller den funktion som den påstår sig göra i teorin. Lundgren går så långt att han menar att det är ett säkert sätt att misslyckas när man implementerar hela XP. Detta påstående anser vi motbevisas något av Pendleton, då han nämner att det projekt han coachade blev mycket lyckat. De använde XP fullt ut, bortsett från praktiken ”test-first”,på

grund av den omfattning det hade inneburit. Även om hela XP inte användes här användes en mycket stor del, vilket kan tyda på att en praktiskt tillämpning av hela metoden faktiskt skulle fungera. Dock kan inte den slutsatsen dras baserat på detta material.

Enligt Mirakhorli et al., (2008) och Fitzgerald et al., (2005) är det inte alla projekt som drar nytta av hela metoden vilket medför att den måste skräddarsys med tanke på de omständigheter som råder. Det styrker vårt antagande kring skillnaden på den akademiska sam näringslivets syn på XP – akademikerna väger mot att metoden är komplett och bör användas fullt ut medan näringslivets syn är över lag är den att metoden fungerar bäst i anpassad form och eller i kombination med någon annan systemutvecklingsmetod.

När XP anpassas säger alla tre informanterna med praktisk bakgrund kring metoden att XP och Scrum fungerar bra ihop. Vidare nämner de även att Scrum bidrar med projektstyrning och XP med den tekniska delen. Dessa två påståenden får även medhåll från Fitzgerald et al., (2005), som poängterar att Scrum och XP fungerar bra tillsammans av just denna anledning. Bendix anser att anledningen till att Scrum används mest i arbetslivet idag beror på de certifieringar man kan erhålla inom metoden.

Lundgren drar paralleller mellan XP och en verktygslåda, han menar att i praktiken tar de utvecklarna de verktyg de behöver. Han menar att processen är omvänd. Istället för att utgå från XP och ta det som är önskvärt, så plockas tekniker från olika källor för att sedan visa sig tillhöra XP. Ett fenomen som skulle styrka att XP lyckats med att hitta fördelaktiga lösningsmetoder, även om alla praktiker inte lockar alla användare. Beck, 2000 hävdar att de tolv praktikerna understödjer varandra samtidigt som Mirakhorli et al., (2008) nämner att det är svårare att se förhållandet mellan praktikerna än praktikerna själva. Detta anser vi pekar på att metoden är enkel att förstå, men svår att nyttja korrekt i praktiken. Det krävs antagligen erfarenhet för att värdesätta metoden, vilket även Bendix påstår. Detta påstående anser vi får medhåll från Lundgren, då han säger att man måste köra XP ett tag innan man kan se om det fungerar. Vidare nämner han att det faktiskt är där vi är idag, vi ser att det fungerar. Vi frågar oss själva om hur det fungerar för de praktiska brukarna av metoden idag, genom en blandning av metoder, eller om metoden idag fungerar så som Beck utformat den. Det som pekar på det första alternativet är att Fitzgerald et al., (2005) menar att XP är en lättviktsmetod, som man kan plocka delar från för att komplettera den befintliga agila metod som används.

När vi har tittat på praktikerna var för sig har vi funnit ett antal som verkar mer omdiskuterade än andra. Dessa är framför allt customer-on-site, parprogrammering samt testdriven utveckling. Gällande customer-on-site så skiljer uppfattningarna mellan informanterna åt. Bendix menar att kunden är viktig att ha på plats, framför allt på ett kort avstånd från utvecklarna, så att de inte tar beslut åt kunden. Lundgren har däremot aldrig använt sig av customer-on-site och har ingen åsikt kring fenomenet. Mellan dessa ytterligheter befinner sig Lundström, som använder customer-on-site, dock utan större krav på avstånd mellan kund och utvecklare. Problematiken kring denna praktik menar Rumpe & Schröder, (2002) är huruvida man kan övertala kunden om nyttan med att ha en representant ständigt närvarande. De menar också att detta är en av de praktiker som upplevs som mest problematiska att uppfylla. Ett bra resonemang från teamet krävs för att övertyga kunden om att spendera en heltidstjänst som kan bistå utvecklarna under hela projektet, alltså kan detta ses som en ekonomisk fråga utifrån kundens perspektiv. Karlström, (2002) och Hussain et al., (2008) menar dock att i de fall då kunden har deltagit så har resultatet fallit väl ut, vilket pekar på att problemet ligger i att teamet behöver föra fram behovet för kunden på ett bättre sätt.

Parprogrammering är däremot informanterna mer överens om. Bendix anser att parprogrammering är onödigt då enkla uppgifter skall utföras och att det då istället räcker med kodgranskning. Lundström har nästan samma uppfattning, att parprogrammering fungerar bra ibland, när man saknar specifikationer eller vill sprida kunskap. Han menar att man tar det när det behövs, i korta intervaller. Kunskapsspridningen är även Fitzgerald, Harnett & Conboy, (2006) inne på. Lundgren gillar parprogrammering men använder det inte alltid. En del i problematiken med denna praktik skulle kunna vara en resursfråga. Bendix menar att man inom näringslivet ser på parprogrammering som att det är två människor som gör en persons uppgifter, och det är fel anser han. Möjligen är det så att beställaren ser praktiker som resursslöseri och därför har svårt att acceptera den. Källan i problematiken skulle kunna ligga i Kent Becks något idealistiska syn på metoden. Tanken och utfallet av praktiken är mycket gott, men svårt att sälja in till den ekonomiansvarige i projektet.

Testdriven utveckling anser Lundgren vara bra, men Lundström uttrycker det som att man ej blir produktiv om man fastnar i det. Problematiken med praktiken uttrycker även Hedin, Bendix & Magnusson, (2003) genom att poängtera svårigheten med att veta på vilken nivå tester ska ske, vilket kräver tydliga exempel för att praktiken ska bli lättförstådd.

Kontext

Många delar av kontexten har vi behandlat under övriga delkapitel här i diskussionen och därför hanterar vi här endast de som inte innefattas av dem.

Systemutvecklingens kontext är flitigt nämnd inom litteraturen. Tyvärr är det en otydlig bild av kontexten som målas upp eftersom det saknas en allmänt spridd definition. Försöken som gjorts med att försöka fånga in vad kontexten består av har syftat till att underlätta arbetet med att undersöka hur den påverkar utvecklingsarbetet. Dock har det visat sig vara ett komplext arbete. Att veta vilka delar i kontexten som är relevanta är inte sällan svårt och diffust. I Fitzgerald et al:s (2002) definition är kontexten en separat del bland till exempel utvecklare. Ramverket som Williams et al (2004) presenterar placerar istället utvecklaren som en del i kontexten. Denna problematik stöter vi på hos informanterna då frågan uppkommer ”Vad menar de med kontext?”. Lundgren nämner att de praktiker som behövs plockas utefter den kontext man befinner sig och Bendix är inne på samma resonemang. Han säger att kontexten påverkar vilka delar av XP som väljs och det är ett genomgående tema hos informanterna. Problemet ligger i att kontexten inte blir en konkret, mätbar parameter och att då använda ordet i ett resonemang utan någon förklaring eller definition kan lätt leda till felaktiga antaganden. Vad kontexten är blir godtydligt och resultaten diffusa. För att hantera detta problem har vi valt att konkretisera kontexten i olika *beståndsdelar* baserat på akademisk litteratur. De stora beståndsdelarna, och det som vi har valt att koncentrera oss på har därför tagits upp tidigare i diskussionen och därför nämns kort några andra delar av kontexten som informanterna har tagit upp.

För att då gå ner på en mer konkret nivå har informanterna nämnt olika saker gällande hur kontexten som påverkar utvecklingsarbetet. Pendleton nämner att landets kultur spelar in när det gäller att anamma det agila tankesättet. Han menar på att det till exempel i Sverige är lättare att implementera XP-metoden än i Danmark. Ett sådant resonemang passar in på kategorin ”geografiska faktorer”. När det gäller ergonomiska faktorer så har Bendix talat om de svårigheter som uppstår när kunden inte kan finnas på plats i projektet. Användningen av en praktik som parprogrammering blir beroende av den miljö som teamet befinner sig i, då de behöver kunna göra sig förstådda.

Det kan tyckas självklart att de rådande omständigheterna i projektet påverkar metodanvändningen men då XP är tänkt att vara en heltäckande metod är detta anmärkningsvärt. Dock måste återigen diskussionen om kontextens definition beaktas här. XP-metoden förutsätter att dess praktiker kan utföras av utvecklare samt att de får möjlighet att göra det i organisationen. Det är därför tydligt att dessa två delar, som ingår i kontexten, gör XP-metoden beroende av vissa kontextuella förutsättningar. Då Jones (2000) dessutom nämner över 250 stycken faktorer i kontexten som påverkar utvecklingsarbetet kan inte alla faktorer undersökas här, och det är kanske inte rimligt att alla dessa skulle kunna hanteras av en metod. Följaktligen borde alla metoden vara kontextkänsliga, det är bara en fråga om hur mycket.

Användbarhet

I den definition på användbarhet som Bevan et al., (1991) använder menar de att lättanvändhet i en viss kontext innebär användarens prestation och tillfredsställelse av själva tillämpningen. Den praktiska synen på XP i detta fall är enligt Bendix den snabba feedback och gången i projektet. Detta påstående får medhåll från Pendleton som uttrycker det som att teamets kommunikation ökar, vilket främjar bra lösningar och ett kollektivt tänkande. Vidare talar Pendleton om att XP ger en medvetenhet i teamet om vad de andra gör, vilket saknas i traditionella metoder. Detta är direkt kopplat till användbarhet då Bevan et al., (1991) talar om att ett nytt system bör vara minst lika effektivt och tillfredsställande som den metod eller tillvägagångssätt den skall ersätta. Lundström är inne på samma spår och säger att XP ger mer mjukvara med högre kvalitet av färre personer på kortare tid än traditionella systemutvecklingsmetoder. Detta menar Hartson, (1998) är användbarhet, då snabbheten med vilken användaren når sina mål är en viktig komponent i definitionen på användbarhet.

Problematiken med XP:s användbarhet beskriver Pendleton då han nämner att metoden inte fungerar därför att den inte kommer upp på dagordningen på möten där val av systemutvecklingsmetod eller process beslutas. Det är på dessa möten som makten och pengarna finns enligt Pendleton. Thong et al., (2004) påpekar att uppfattningen av en produkt är av betydelse för den fortsatta användningen. Här ser vi en problematik med den praktiska användningen av XP, då metoden på något sätt inte når tillräckligt högt i företagshierarkin och möjligtvis inte visar på sin användbarhet för rätt individer på företaget. En annan syn på denna problematik kan vara det som Pendleton hävdar kring projektstyrningsdelen i XP, att den metoden inte adresserar tillräckligt mycket kring projektstyrning. Detta hänger ihop med den upplevda användbarheten för den individ som tar beslut på företag kring vilken systemutvecklingsmetod som skall användas samt den upplevda användbarheten för organisationen.

Att metoden anpassas är en relevant del av användbarhetsdiskussionen. Pendleton menar att vissa praktiker skulle kunna brytas ut och tillämpas i andra systemutvecklingsmetoder. Van Weile (2001) menar att användbarhetsproblem i grunden beror på en missanpassning mellan de färdigheter som användaren besitter och de färdigheter som systemet kräver av användaren. Det kan vara så att XP i sin helhet kräver färdigheter av användaren som denna inte har. Således blir enstaka praktiker användbara i andra metoder, men sammantaget blir XP svårt att använda. En möjlig tolkning av detta är att metoden är framtagen av en skapare som har en akademisk bakgrund och därmed har en större tro på formaliserade metoder. Att Bendix som också befinner sig i den akademiska världen är av samma uppfattning som Beck – att metoden skall användas i sin helhet tyder på en skillnad mellan den upplevda användbarheten för näringslivet respektive inom den akademiska världen.

Att en systemutvecklingsmetod är tänkt att hjälpa användaren att nå sitt mål och därför behöver vara användbar för den tänkta målgruppen belyser Lundgren då han påpekar att när metoden är i vägen för det dagliga arbetet tappar den sitt värde. Han säger även att det inte är något självändamål att följa metoden.

6.1 Sammanfattning av de viktigaste iakttagelserna

Av vår undersökning framgår att XP-metoden inte är problemfri att använda. Det stora antalet praktiker som skall tillämpas fullt ut innebär att metoden framstår som svår och kravtyngd. Kent Becks bakgrund som utvecklare genomlyser metoden och den uppvisar bristande användbarhet för organisationer, då den saknar tillräckligt med stöd för projektstyrning. Vidare är vissa praktiker svåra att sälja in, exempelvis customer-on-site. Även XP-metodens bristande förståelse för organisationens krav på metoder och processer innebär att metoden ofta säljs in underifrån, från utvecklarna, snarare än att upplevas som ett överlägset tillvägagångssätt av de som befinner sig högre upp i organisationens hierarki. Att förekommande metodanpassningar ofta uppskattas högre av utvecklarna än användandet av metoden i sin helhet, är ett tecken på att den i sitt grundutförande inte framstår som tillräckligt attraktiv och användbar av många organisationer och utvecklare.

Vidare ställer XP-metodens krav på utvecklarna till problem då flera olika kunskapsdomäner behöver bemästras så som test och arkitektur. Visserligen är många, speciellt seniora utvecklare kreativa men denna kreativitet anses snarare bidra till att kompensera för metodens brister än att tillämpa praktikerna till punkt och pricka på den aktuella uppgiften. Detta kan anses vara en svaghet hos metoden.

När det gäller att leverera en effektiv process finns det däremot en hel del underlag som visar på att så är fallet. Många studier som granskat XP vittnar om en framgångsrik metod som levererar resultat, oavsett om den används fullt ut eller är anpassad. Således kan XP upplevas som användbar både för organisationer och för utvecklare då den faktiskt kan leverera värde.

Kontexten som helhet verkar spela en roll vid bruk av XP då vissa miljöer och uppgifter verkar lämpa sig klart bättre än andra. Vad som är avgörande här är dock väldigt diffust och att i förväg förutsäga användbarheten av XP i en viss kontext är svårt.

6.2 Avslutande diskussion

Efter att ha gjort en förstudie till uppsatsen och då börjat studera teori om XP framträdde en bild av en genomtänkt metod som skulle fungera väl i praktiken.

När det senare visade sig vara problematiskt att finna informanter skapades ett intresse om att ta reda på varför det var så svårt att hitta personer som arbetat med XP. Vid fördjupade efterforskningar hittade vi mycket information kring svårigheter med att använda XP samt att metoden ofta anpassas. Att flera informanter även gav bilden av XP som problematisk vid användning, gjorde att de förväntningar som skapades under förarbetat inte visade sig stämma. En möjlig förklaringsmodell är att de artiklar om XP som gavs ut nära i tiden efter metodens lansering belyste just användningen av XP fullt ut och hur det hade fungerat, varav flera visade på ett positivt resultat. Längre fram i tiden kunde fler och fler röster urskönjas som tog upp problematiken med metoden för att sedan fortgå med olika former anpassning.

En rimlig förklaring skulle kunna vara att XP-metoden i början endast användes i begränsad omfattning. När den senare började användas i större omfattning i en mängd olika kontexter framkom fler och fler påpekanden om metodens brister.

Vi kan grovt urskilja två olika läger hos informanterna. Bendix och Pendleton menar båda på att hela XP kan användas och att metoden fungerar bäst när så görs. Visserligen pekar de på svårigheterna med att genomföra en fullskalig tillämpning men deras grundinställning är ändå att XP kan fungera som det är tänkt. Detta står i kontrast till det som Lundgren och Lundström uttrycker. De menar att det inte fungerar att använda XP så som det är tänkt. Intressant i sammanhanget är att de sistnämnda är de som uteslutande använt sig av XP rent praktiskt medan Bendix och Pendleton haft en mer observerande och teoretisk roll. Möjligen skulle de som arbetat praktiskt har en större insyn i hur XP fungerar i praktiken, men samtidigt har de informanterna aldrig uttalat försökt följa hela XP och därmed kan ingen sådan slutsats dras. Men en anledning till att de inte har följt hela metoden skulle kunna vara att de upplever brister med metodens användbarhet.

Vi har i den yttre validitetsfrågan eftersträvat att från vårt mindre urval av informanter försökt överföra slutsatser till en mer teoretisk nivå, eftersom att vi ha använt oss av kvalitativa intervjuer. Att överföra slutsatserna till en större population från vårt urval anser vi inte vara möjligt utifrån de förutsättningar vi hade rådande antalet informanter och deras bakgrund. Gällande den inre validiteten har vi efterstävat intersubjektivitet, att nå så nära sanningen vi kan komma. Vi är medvetna om att vårt resultat inte automatiskt *är* sanningen. Men för att försöka komma så nära sanningen som möjligt har vi granskat resultatet kritiskt och jämfört det mot andra källor.

För att uppnå tillförlitlighet i vårt resultat har vi eftersträvat replikerbarhet. Genom att hålla transkriberingarna intakta och redogöra för vårt tillvägagångssätt har vi gjort det möjligt för andra att dra slutsatser utifrån vårt material. Med nya informanter är möjligheten stor att resultatet kommer att se annorlunda ut eftersom att urvalsgruppen av informanterna kan uppleva XP på olika sätt. Detta medför att vårt resultat är svårt att generalisera och att vi därför har eftersträvat mer bredd än djup på uppsatsen.

Problematiken med att hitta flertalet informanter som arbetar praktiskt med XP medförde att vi fick koppla varje informants svar till delkomponenterna i vårt ramverk och föra en diskussion kring varje del. Detta gjorde vi för att öka generaliserbarheten, då de olika informanterna har fått olika intervjufrågor att svara på, vilket gör det svårt att direkt se skillnaderna mellan varje informant. På så vis får vi en ökad bredd, men minskat djup på vår empiriska del. Vi önskar att man i framtida studier kring XP enbart intervjuar en typ av brukare, som till exempel personer som har arbetat praktiskt med metoden. Vi reflekterade även över problematiken med att ställa samma intervjufrågor till dem, trots deras olika erfarenheter av XP.

Frågan kring vem man ska intervjua kring XP kan diskuteras. Då informanterna som var praktiska brukare av metoden framhöll att den sällan når högre än projektledaren hade en empirisk studie av dessa projektledare eller till och med intervjuer med individer ännu högre i hierarkin varit intressant att ta del av. Eftersom att vi hade problem med att finna informanter har vi funderat över tillvägagångssättet, att man istället för som vi gjorde, först göra en litteraturstudie och sedan gå ut i arbetslivet bör man kanske gå andra hållet. Att först kontakta några företag och där se vilken eller vilka systemutvecklingsmetoder de använder, för att sedan fördjupa sig i dessa teoretiskt. Då blir uppsatsen mer relevant för arbetslivet, men möjligtvis mindre intressant ur en akademisk synvinkel då fokus ligger på praktiskt bruk. Vilket av dessa sätt som är mest relevant är tämligen svårt att svara på, men en reflektion

kring den akademiska trögheten är intressant. Något som är aktuellt i arbetslivet kanske kan ta lite tid att nå studenterna, men å andra sidan skapas många metoder och tekniker teoretiskt innan de brukas praktiskt, varför akademisk forskning i vissa fall ligger i framkant av vad som är aktuellt för tillfället.

Vi har funderat på hur uppsatsen skulle kunna förbättras och kommit fram till att en avgränsning kring informanter eller ett mindre forskningsområde hade begränsat uppsatsens omfång. Att enbart granska användbarhet för utvecklaren och då enbart intervjua en viss typ av individer hade gett ett ökat djup och styrka åt generaliserbarheten. En jämförelse av ett sådant arbete med denna uppsats hade varit intressant.

7 Slutsats

Syftet med denna uppsats var att finna svar på vår frågeställning som lyder:

Hur upplever systemutvecklarna samt deras organisation XP-metodens användbarhet i praktiken?

Resultatet visar på svårigheterna för systemutvecklare att använda metoden i sin helhet. Den ställer stora krav på en bredd hos utvecklaren men brister även i riktlinjerna för hur metoden skall användas konkret. Detta leder till att utvecklarna upplever praktikerna i metoden som svåra att förstå, samt tillämpa dem som det är tänkt. Själva användningen av metoden är inte helt problemfri då praktikerna inte används så som det är tänkt. Det positiva är ändå att XP har visat sig leverera resultat, det vill säga att systemutvecklarna når sina mål. Även produktivitet och prestationsförmåga ökar vid användning av XP-metoden. Utvecklarnas acceptans för metoden är varierande. I sin helhet har metoden inte stort stöd, men i en anpassad version framstår metoden som attraktiv och användbar för utvecklaren.

Organisationer har en mer kritisk inställning till metoden och XP får inte lika stort stöd där som hos utvecklarna. Metoden riktar sig inte lika tydligt till organisationer och lyckas inte sälja in hur den kan hjälpa organisationen att nå sitt mål. Visserligen visar XP på att kunna leverera ett effektivt arbetssätt och nå resultat, men kräver samtidigt en hel del av organisationen. Detta bidrar till att metoden är mer svårförstådd för organisationer och därmed har de inte lika lätt att ta den till sig. Många organisationer vill anpassa XP-metoden för att den skall ses som användbar i deras kontext.

XP:s användbarhet visar även på att kontexten påverkar huruvida metoden upplevs som användbar eller inte. Vilken typ av uppgift som skall utföras, vilka utvecklare som skall utföra dem samt vilken organisation där utvecklingen skall ske i visar sig alla vara variabler som påverkar.

Vår undersökning visar att de största bristerna med avseende på XP:s användbarhet är:

- Bristande projektstyrningsstöd samt ställer höga krav på organisationen
- Svårt för utvecklarna att tillämpa enligt anvisningarna samt kräver stor kunskapsbredd
- Användbarhet varierar beroende på kontext

Bilagor

B1) Intervjufrågor till informanterna

Frågor till informant 1: Christian Pendleton

Personbeskrivning
- Hur ser din bakgrund ut?

Typ av XP-projekt
- Vad var det för typ av projekt?
- Hur många arbetade i projektet?
- Vad var din roll?
- Hade någon i teamet arbetat med XP tidigare?
- Vet du hur stor del av XP som användes?

Resultatet av XP
- Hur blev resultatet? Var det fördelaktigt att använda XP?

Positiva och negativa aspekter av XP
- Vad upplevde du som positivt med XP?
- Vad upplevde du som negativt med XP?

XP i jämförelse med andra systemutvecklingsmetoder
- Hur upplever du XP i förhållande till andra systemutvecklingsmetoder?

Frågor till informant 2: Lars Bendix

Personbeskrivning
- Hur ser din bakgrund ut?

XP:s popularitet
- Är XP populärt nu?
- När upplevde du att XP var populärt?

XP:s styrkor och svagheter
- Vilka är XP:s styrkor?
- Vilka är XP:s svagheter?

Anledningen till XP:s position i dagsläget
- Vad är anledningen till att du upplever XP som starkt eller svagt?

Metodanpassning
- Sker det mycket anpassning av XP?

XP i jämförelse med andra systemutvecklingsmetoder
- Hur upplever du XP i förhållande till andra systemutvecklingsmetoder?

Frågor till
Informant 3: Conny Lundgren
Informant 4: Thomas Lundström

Personbeskrivning
- Hur ser din bakgrund ut?

Teoretisk kunskap kring XP
- Hur mycket teori om XP kan du?
- Känner du till mer än vad praktikerna innebär?

Praktisk kunskap kring XP
- I vilket sammanhang arbetade du med XP?
- Hur stor del av XP använde du?
- Vilka delar av XP tycker du fungerar bra att använda?
- Vilka delar av XP är svåra att använda?

Positiva och negativa aspekter av XP
- Vad tycker du är positivt med XP?
- Vad tycker du är negativt med XP?

XP:s krav på utvecklare och organisation
- Vilka krav anser du att XP ställer på utvecklarna?
- Vilka krav anser du att XP ställer på organisationen?

B2) Intervju med informant 1

1	Björn	Vi kan ju börja med en grundläggande fråga; vill du vara anonym eller är det okej om vi anger vem vi har intervjuat?
2	Pendleton	Ni kan ange mitt namn.
3	Björn	Ok. Din bakgrund, hur ser den ut?
4	Pendleton	Hmm. Jag pluggade på data på LTH 1987-1994. Jag var väl inte särskilt metodikintresserad då men sen så började jag på företag 1 och började använda configuration management från dag 1. På det företag 1 flyttade jag upp till Stockholm och jobbar man som CM så blir man naturligt processintresserad och processorienterad i sitt arbete för det hör så himla intimt ihopa.
5	Gustav	Okej.
6	Pendleton	Men jag jobbade på det företag 1 i tre år och tre olika verksamheter kan man säga att jag passerade på den tiden och det sista jag gjorde var när de startade om efter nedläggningen av ett några sammansatta företag, för då fick vi göra en ny utvecklingsmetodik till det projektet som skulle starta efter det och då fick jag ansvar för CM-området inom den metodiken. Det var riktigt kul, så då blev jag lite mer intresserad och sen så har jag varit metodintresserad sen dess. Sen började jag konsulta sen 97 på företag 2, bytte till ett annat företag och startade en konsultavdelning 2000. Vad hände sen? Sen dog konsultmarknaden så jag fick sätta mig på kontoret.
7	Björn	Vid it-bubblan?
8	Pendleton	Ja, precis. Och så satt jag faktiskt och koda lite och lärde mig java, satt och hacka lite och sen så 2004 så fanns det efterfrågan igen så då så hamnade jag på ett uppdrag på företag 1 igen. 2005 flyttade jag ner till Skåne igen och började jobbade på företag 3, som strategisk CM och tog fram en utvecklingsmetodik som vi implementerade där och körde i två projekt och som blev väldigt lyckad. Sen så lade företag 3 ner så då så var jag en sväng på företag 4 men hittade inte riktigt rätt där så jag började på Softhouse i februari i år.
9	Gustav	Så du har hoppat mycket?
10	Pendleton	Ja, jag har skuttat lite, mina kompisar sa till mig nu senast att byter du jobb igen, ditt CV ser ju snart ut som Krig och Fred. I natt så drömde jag faktiskt om att jag bytte jobb igen, så där ångestskapande, för jag har ingen lust med det för jag trivs väldigt bra på Softhouse. Agila metoder kom jag i kontakt med första gången när Agile Sweden startades, ungefär i den vevan, uppe i Stockholm 2002, tror jag officiella startåret är. Arbetet började 2001, men vi var med på startmötet i alla fall, några stycken. Det var mycket roligt; vi mötte en helt annan värld, för då kom vi liksom från den här företag 1-världen med stora telekom-projekt på 200-500 personer, mellan två och åtta länder, som pågick i flera år. Så mötte vi då den mer agila världen, det var väldigt mycket ensam-konsult, eller små-konsult, få-manns-konsultföretag som satt och utvecklade web-applikationer med kunden eller två kunder, och förutsättningarna för de projekttyperna är liksom så totalt olika och det var jättekul att möta dem och vi märkte liksom att de här kulturerna, dem, det är ju som att det är helt olika branscher.
11	Gustav	Hur var det på företag 1, var det nån typ av vattenfallsmetod eller liknande?
12	Pendleton	Ja, stenhårt. Det är det fortfarande, mycket.
13	Gustav	Ja, okej, det kanske lite lugnare inom företag 1 då för de har sina egna

		plattformar, det är väl inte lika..
14	Chrisitan	Nej. Kraven svänger där också. Så sitter man, mitt förra uppdrag var på företag 1 här uppe och då satt jag med kravhantering på deras referensapplikationer och skötte det. Och nä, företag 1 har förändrade krav hela tiden. Just på referensapplikationerna gör man kanske inte så mycket, för det ingår inte i kärnprodukten, men jag var med i systemgrupperna liksom och såg hur mycket det svänger hela tiden. Massor med change-request-hanteringar.
15	Gustav	Agile Sweden 2002, vad var det för olika metodiker de tog upp där?
16	Pendleton	Alltså, det som var då, som alla pratade om då var XP. Det var den första som liksom som anammades och som var stor då, och Kent Beck var lite sådär gud, alla pratade om honom. Det var så fräckt med parprogrammering, kontinuerlig testning, det var omvälvande. När man möter de agila, eller…, när man möter topparna i agil-sammanhang så pratar de så väldigt mycket om att det är en revolution, att det är en sådan omställning. Jag är rätt säker på att det också är det, i stora delar av världen, men inte lika mycket i Sverige, och kanske i Norge. För i Sverige och mycket i Norge…, det är annorlunda i Danmark och Finland faktiskt, så har man redan ett rätt så distribuerat arbetssätt, man har ansvar på rätt så låg nivå. Det är hyffsat högt i tak ändå, det är inte så mycket toppstyrning som man möter. Kör man projekt som är multikulturella så märker man att, det räcker att man går till Danmark, kommer man då till Sydeuropa så är där helt annan mentalitet. Där är chefen som bestämmer allt, möjligen kan han delegera till projektledaren, som får vara med att bestämma lite, men allting skall liksom upp till chefen och vända hela tiden. Man tar inte några beslut på låg nivå. Där är det ju revolution med agilt tankesätt och informationstransparens.
17	Gustav	Är det svårare där till exempel i Danmark, det har man ju förstått, att det är väldigt mycket chefstyrt?
18	Pendleton	Jag vet faktiskt inte hur svårt det är där, för jag har inte varit där och konsultat alls. Jag har inte jobbat där, men vi har diskuterat kulturskillnader en del och det är skillnad i Danmark. Det är inte som att komma ner till Italien eller komma över till USA, men det är en viss skillnad på kultur.
19	Gustav	Det var bra du sa det, för vi ska intervjua en dansk lektor, så då får vi passa på att fråga honom det. Vill du berätta lite kort om CM, vi känner att vi inte riktigt kan det.
20	Pendleton	Ja, jag håller en tredagarskurs…
21	Gustav	Haha, ok, men bara så att vi kan förstå lite.
22	Pendleton	Alltså, CM handlar om att hålla ordning och reda, det är huvudparadigmet. Man skall ha spårning, och kontroll, och lite så. Man brukar bryta ner det i fyra subdiscipliner, traditionellt sätt. Identifiering, där man skall kunna sätta en identitet på varje fel eller dokument, inklusive en version då. En identitet och en version skall finnas på varje sak man vill kunna hantera. Man bestämmer sig för en den högsta abstraktionsnivån. Ofta när man pratar om dokumentation så är det ett dokument. Pratar man om källkod så är det väl rätt vanligt att man klumpar ihop, här har vi ett block eller så och då är det de här 5-8 källkodsfilerna kanske. Sen så pratar man om Change Management, som då är ändringshantering, man bestämmer sig för att frysa någonting, eller så baseline, det är ju begreppet inom CM-världen, och det handlar egentligen om publicering, alla är överens om att det är det här som gäller, det här gränssnittet ska vi jobba mot och vi har två grupper som skall finnas, och de skall kommunicera, att det finns kommunikation mellan det de utvecklar, och så måste de ha ett gränssnitt mellan sig. Fryser man gränssnittet att de kan jobba parallellt sen mot det här

		gränssnittet, de utvecklar en funktion som de skall använda och så har de någonting att jobba genom. Krav-baseline är kanske det vanligaste, att man fryser kraven för projektet liksom. Frysningen leder då till att man sen måste ha en formell hantering på förändringar av den här frysta.
23	Gustav	Om de skall förändras så behöver de ha en…
24	Pendleton	Exakt. Kör man vattenfall så har man en typisk krav-baseline i början och sen så gör man ändringhantering mot den. Folk kommer väl ändra kraven, ja, då lägger man en change-request och så går den igenom beslutsforum, analys och så vidare, så har man en process för det, med statuskedja och grejer och så tillslut så förändrar man kravet förhoppningsvis eller så säger man nej, så lägger man det på vänt, eller vad man nu bestämt sig för.
25	Gustav	Skapa mer ordning och reda i processen?
26	Pendleton	Ja. Det här är ju.. Just det, de andra disciplinerna. Configuration-control som handlar om statushantering för configuration-item som man sätter vid identifieringen. Vad är det för status på produkten egentligen, hur långt har vi kommit, vilka versioner av systemet är det vi har i test, vilka har vi i utveckling, vilka spår har vi öppna med mera. Branchning i versionshanteringen och hur den skall skötas och så.
27	Gustav	Så det är mycket mot projektledare eller någon i lite högre position som håller koll?
28	Pendleton	Ja, det är ju en stödfunktion egentligen och jag brukar försöka fokusera på att stöda utvecklarna så mycket som möjligt, man får mest pang-för-pengarna då tycker jag. Men det är klart, att den här ordningen och redan skapar man ju för att kunna optimera verksamheten på något sätt, skapa bättre förutsättningar för kvalitet och korta tiderna och kunna jobba snabbare, kortare iterationer eller snabbare feedback och så.
29	Gustav	Så ett av dina säljargument är att du effektiviserar?
30	Pendleton	Ja. Och den sista subdisciplinen, configuration-audit, är någonting som går lite upp och ner. En del företag jobbar väldigt mycket med det, andra gör det inte alls, så det är väldigt olika hur man jobbar med det, men nån typ av konfigurationsgranskning . Det är rätt så hårt verktygsfokus på den sidan. Innan jag började jobba fanns det inte mycket CM-verktyg, då satt man mycket med listor med produkter och jämförde om man hade samma version på båda papperna, samma som vi har levererat som vi har testat, och det behöver man inte göra nu utan nu skriver man ett script som kör igenom listan.
31	Gustav	Det låter som det här skulle vara väldigt populärt inom flygindustrin?
32	Pendleton	Ja, det är det. Jag håller CM-kursen några gånger om året och de två senaste gångerna som vi har kört nu i höst har ett företag fyllt båda kurstillfällena.
33	Gustav	Ok, det har dom. Det kan jag tänka. Agilt arbetssätt… hur var det nu 2002, du sa att XP var populärt och att Kent Beck var lite gud, det gav så att säga en bild av att det var väldigt stort.
34	Pendleton	Nä, inte stort. Det var nytt och fräscht, omvälvande och nytänkande. Utvecklare tycker för det mesta inte om att jobba på det sätt man oftast gör i stora organisationer där man har, det är mycket arbete i en stor organisation som inte handlar om att skriva kod, som inte är kreativt arbete. Utan det är mer läsa dokument, sätta sig in i problemen, analysera vad som gjorts tidigare, och sen har man sitt kreativa kodskrivande, men sen så ska man dokumentera vad du har gjort så att någon annan kan jobba vidare med det sen och sen får du ändå tillbaka din kod från test, där de säger att det här är bara skit, det får du skriva om, eller det är en massa fel och grejer. Kent Beck säger att dokumentation bryr man sig inte om, det som är viktigt, det som faktiskt leder fram till att vi får en fungerande produkt, det skall vi göra mycket. Så testa är bra, det kontrollerar vad vi

		gjort, det ska vi göra hela tiden. Att man granskar varandras kod är viktigt, det har vi sett ökar kvaliteten, det ska vi göra hela tiden, hänger ihop med parprogrammeringen. Vad är det mer han har? Att planera långt i förväg är dåligt, det sysslar vi inte med utan vi ska ha små uppgifter som man kan jobba dynamiskt med, och ha ständig förändring. Det som kom i början på 2000-talet var en blandning, dels XP som var mer handfast; så här ska man jobba, men det var ju också de här agila principerna som på nått sätt vävdes in, och jag fick inte någon riktig bild då i alla fall vad det var som var vad, vad var XP och vad var det andra, utan det kom på något sätt som en revolutionstanke. Kom ungefär samtidigt som folk började prata om patterns i en större utsträckning, det var väldigt mycket metodsnack och det var ju jättekul, för det kändes nytt och fräscht, energifull och roligt.
35	Gustav	Så man kan säga att den agila metoden var någon sorts vision och XP var mer hands-on i praktiken?
36	Pendleton	Ja, alltså, det Agile Manifesto handlar mycket om värderingar, och de olika metoderna där tar ju de olika värderingarna där. De säger att för att vi ska kalla oss agila ska vi ställa upp på värderingarna i Manifesto. Sen kan man göra det på olika sätt, och XP har ju valt en väldigt utvecklar-nära implementation. Ska man sen vara elak mot XP så kan man ju säga istället att det var några knasiga small-talk programmerare i USA som satt och tänkte och inte tyckte om dokumentation och inte tyckte om projektledare, så de ville bestämma själva, så de hittade på en metodik och deras första projekt gick helt fel men de fortsatte i alla fall.
37	Gustav	Gick det helt fel?
38	Pendleton	Ja. Första projektet blev nerlagt.
39	Gustav	Okej, kanske inget man har skrutit med då nä.
40	Pendleton	Det står på wikipedia.
41	Gustav	Vi får inte använda den källan, då blir vi strypta.
42	Pendleton	Haha, okej. Fast när det gäller agila metoder är de nog ganska pålitliga, finns rätt mycket info där. Men man måste kanske verifiera den och hitta den riktiga källan sen.
43	Gustav	Då har man fått idén i alla fall. Vi går vidare…
44	Björn	Ja, vilken typ av projekt var det som du coachade?
45	Pendleton	Ja, det var ett projekt på företag 1 faktiskt. De skulle byta processor i en telefonväxelstation, från sina egenutvecklade processorer till ett annat märke på processor, jag tror att företag X äger dem nu. Parallellt med det så ville man kolla om det gick att porta den till en tredje processor samtidigt. Då var det ett konsultgäng uppe i Stockholm som jag hade jobbat med tidigare i ett annat projekt. Det projekt som de satt med blev nerlagt, så då blev de lediga och då såg man att de hade jobbat med liknande grejer innan, otroligt duktiga. En av dom var rabiat CM-hatare, och har dessutom Aspbergers-syndrom, utpräglat, och har problem med att kommunicera med folk på ett normalt sätt. Han är otroligt skärpt och kan man ta honom så har man mycket ut av honom, jag respekterar honom och han respekterar mig. Så när de skulle starta det här projektet så var det så att företag 1 kräver att vi skall ha en CM även om vi bara skall köra 8 pers. Så vem ska vara CM? Det får han som hatar CM vara. Ja okej, men han kanske behöver lite hjälp, då kan jag coacha honom, så jag var med lite i periferin, som en mentor till honom. Så jag var med på några projektmöten och stod med på deras mailinglistor, och såg vad som hände.
46	Björn	Så du var inte i projektet så att säga?
47	Pendleton	Nä, jag var lite coachande vid sidan om, till hans CM-arbete då.
48	Gustav	Aha, okej. Då frågar vi lite om XP, det verkar som om du har mycket mer koll på det än vad vi vågade tro. Vad var din inblick i projektet när det gällde XP, fick du någon uppfattning om hur mycket av metoden dom

		använde, hur seriösa dom var?
49	Pendleton	Dom var rätt seriösa, för det var en medveten satsning eftersom att det projektet gick ungefär 2002, så vi var jättesugna, det var i alla fall fyra i projektet som hade varit med i det startmötet på Agile Sweden, och
50	Pendleton	Vi var med på Agile Swedens mailinglista för medlemmar allihopa, när det var väldigt mycket skitsnack i och för sig. Eller i alla fall väldigt mycket resultatlöst diskuterande, konsultdöden var precis i faggorna, så det var väldigt många ensamkonsulter som satt hemma och inte hade någonting att göra som skrev inlägg på mailinglistan, kändes det som i alla fall. Så trafiken på mailinglistan var helt sjuk emellanåt, ett av de första grejerna man gick med på var att sätta upp ett filter i Outlook, in med alla mail i den här lådan så får jag läsa det när jag har tid. Men dom var rätt så seriösa, de tog verkligen grundparadigmerna och försökte få in dom ordentligt, och dom lyckades bra.
51	Björn	Använde de XP fullt ut, de blandade inte med någon annan metod?
52	Pendleton	Nä, de körde XP fullt ut.
53	Gustav	Körde de alla de här tolv praktikerna som Kent Beck har satt upp?
54	Pendleton	Ja, nu ska vi se. Det som de inte riktigt klarade var test first. Och det berodde ju på att det fanns, dom skulle ju göra en portning på ett existerande system, så det fanns, alltså telefonväxeln var ju redan färdig, och de skulle göra portningslagret till en ny processor, så om dom skulle skriva testprogram till allt som redan fanns så hade de 20-års arbete framför sig, utan de fick liksom fokusera på det dom själv skulle utveckla, försöka isolera det och skriva testprogram för det.
55	Gustav	Så de skrev test för det dom själva skrev?
56	Pendleton	Ja.
57	Gustav	Men inte för det som fanns?
58	Pendleton	Nej, det kan man väl säga. Det fanns klart ett överlapp, men det var väl den paradigmen dom fick ta.
59	Gustav	Men dom kör parprogrammering och så?
60	Pendleton	Dom körde parprogrammering.
61	Gustav	Okej, det har jag förstått är en av de mest omtalade praktikerna när det gäller XP.
62	Pendleton	Ja, och det är nog det som det finns flest olika lösningar på också.
63	Gustav	Okej. Rent praktiskt eller?
64	Pendleton	Ja, det är många som har försökt med äkta parprogrammering och som har dissat det ganska snabbt, för att man säger att nej, det här är inte effektivt. När man bara skriver den enkla koden är det meningslöst.
65	Gustav	Okej, det är mer när man har ett svårt problem?
66	Pendleton	Ja, och jag sitter ju i ett Scrum-projekt nu i Malmö, och vi använder parprogrammering där också, så när de har problem så kallar dom på hjälp. Vi säger inte på nått sätt att vi använder XP där inne, men den paradigmen är liksom det, behöver man hjälp så får man det och så sitter man tillsammans två och två och tar sig igenom just den problemställningen. Så den approachen tycker jag fungerar väldigt bra. Om man nu inte kör riktig parprogrammering, så åtminstone täta kodgranskningar. Det körde dom i det projektet på företag 1, att dom faktiskt granskade varandras kod rätt så kontinuerligt, med bra resultat, de hade hög kvalitet.
67	Gustav	Så det blev lyckat, detta projektet?
68	Pendleton	Ja, det var ju ett försöksprojekt, och nu minns jag faktiskt inte om företag 1 någonsin gick på den processorn eller om man struntade i det. Men de fick fram en fungerande lösning, dom visade att det gick att porta.
69	Gustav	Okej, så om man bara tänker på att XP lyckades så kan man säga att det gjorde det?
70	Pendleton	Ja, det kan man säga, i alla fall i det fallet.

71	Gustav	Upplevde du att det var fördelar med att använda XP?
72	Pendleton	Dom trivdes ju på jobbet i alla fall. Just det där utvecklingsteamet var lite speciellt, för de hade jobbat tillsammans väldigt mycket innan, och jag tror inte det hade spelat någon större roll vilken metod de hade använt, så hade de lyckats i alla fall.
73	Gustav	De har sina knep för hur de skall göra?
74	Pendleton	De är otroligt duktiga utvecklare och otroligt duktiga arkitekter, och det är svårslaget. Det spelar ingen roll, har du duktiga människor i projektet så spelar det ingen roll vilken metodik de använder för det kommer alltid att bli bättre än någon som inte har lika mycket erfarenhet.
75	Gustav	Så det är ingen gyllene väg till framgång?
76	Pendleton	Nä, det tycker jag inte. Det är viktigare att ha bra människor än att ha en bra metodik. Duktiga människor löser alltid problemen.
77	Gustav	Fortsatte företag 1 att använda XP någonting?
78	Pendleton	Inte i den verksamheten. Telefonväxel-verksamheten är, jag vet inte om jag vill kalla den rigid…
79	Gustav	Lite stel?
80	Pendleton	Lite traditionell får man väl säga, stel i sitt metodarbete, och det är så väldigt mycket uppbyggt kring telefonväxeln, i verktyg och hur systemet är strukturerat, så det ställer metodarbetet i en lite speciellt situation. Jag har varit där inne och konsultat många gånger och det är inte lätt att förändra saker just på företag 1 telefonväxelverksamhet, det är lättare på andra ställen.
81	Björn	Vi har ju redan varit inne på det som är positivt samt negativt, är det någonting annat du vill ta upp?
82		Alltså, jag gillar med många av de agila metoderna att man boostar teamets kommunikation och att man jobbar tillsammans. Det främjar bra lösningar och ett kollektivt kreativt tänkande tycker jag, men även en medvetenhet hos alla om vad dom andra håller på med.
83	Gustav	Så det skulle du vilja säga är kärnan i en sådan här metod?
84	Pendleton	Det tycker jag, och det som jag tycker är viktigast i alla agila metoder, och som väldigt många företag har störst problem med, som man inte heller pratar så mycket om är informationstransparensen. Att alla skall ha tillgång till all information, från marknadsavdelning till utvecklare och testare, alla ska kunna komma åt den information som dom behöver i sitt arbete. Det är en kulturell fråga och ett tankesätt som är svårt att få igenom, särskilt när man börjar blanda in sådant som produktledning och marknadsavdelning. Det är väldigt mycket att marknadsinformationen får ni inte ta del av, den är hemlig och lanseringsdatum och sådant, det vill vi inte dela med oss av. Det är klart, att i ett aktiebolag så finns det lagliga gränser för vad man får lov att släppa lanseringsdatum.
85	Gustav	Inom företaget?
86	Pendleton	Ja, du får inte släppa informationen till någon.
87	Gustav	Inte inom företaget ens?
88	Pendleton	Nej. Om dom skall lansera en produkt, och man väntar sig att lanseringen är kursdrivande på aktien så måste informationen gå ut samtidigt till aktieägare och anställda.
89	Gustav	Okej, så om ett telekomföretag skall lansera en ny mobiltelefon så vet inte de flesta anställda det förrän samtidigt som kunderna?
90	Pendleton	Nej, precis.
91	Gustav	Men blir inte det problem om man kanske ska tajma in vissa satsningar under utveckling eller så?
92	Pendleton	Jo, det är det. Men det är också delvis därför som man måste släppa nyheter om mobiltelefoner ett halvår innan den faktiskt kommer.
93	Gustav	Aha, det är därför. Så man behöver det egentligen för den interna

		organisationen?
94	Pendleton	Ja, det är så uppenbart internt vid det laget, ett halvår innan, vilka telefoner det är som skall komma. Då går telefonprojekten och de involverar kanske 100-150 personer, och har du släppt information till 100 personer om en ny modell, då är nyheten ute, då är den inte hemlig längre.
95	Gustav	Okej, så det har egentligen ingenting att göra med att man vill visa upp sig inför konkurrenterna?
96	Pendleton	Äh, det har det säkert också.
97	Gustav	Okej, men då är jag med. Vi har ju granskat en hel del fakta, läst många artiklar om XP och även Kent Becks bok. Den boken skall vara enkel men…
98	Pendleton	Nä, den är inte så lätt att ta sig igenom.
99	Gustav	Nä. Det som vi har sett är att XP har varit väldigt populärt, precis som du säger, runt 2002, det har skrivits väldigt mycket om det, folk har testat det, vi har sett test-cases bland annat i Lund, LTH har skrivit mycket om det.
100	Pendleton	Ja, dom är rätt så på.
101	Gustav	Vi skall intervjua en på LTH…
102	Pendleton	Vem då?
103	Gustav	Lars Bendix
104	Pendleton	Aha, jag har arbetat tillsammans med honom.
105	Gustav	Du har gjort det?
106	Pendleton	Vi nätverkar ganska mycket, jag och Lars.
107	Gustav	Okej, kanon. Vad häftigt. Det var ju bara positivt att ni har koll på varandra. Anpassning av metoden har vi sett den senaste tiden, folk har sett att XP inte riktigt har levt upp till det dom hoppats eller det har varit för komplicerat eller att det inte har passat och så, och då har det varit mycket snack om anpassning. Man har tagit vissa tekniker eller man har anpassat vissa tekniker. Är det någonting du har upplevt?
108	Pendleton	Nä, inte så mycket med XP faktiskt, men jag har inte tittat så mycket på XP de senaste åren. Vi jobbar ju mycket med Scrum på Softhouse, som projektstyrningsmetod och vi fokuserar inte så mycket på XP där, men vi ser ju att paradigmerna i XP, alltså hur man jobbar i XP, passar väldigt bra in inom en sprint i Scrum. Så om man tar bort alla försök till projektstyrning som XP har, det är inte så himla många, om man tar bort de och ersätter det med det som finns i Scrum och sen tar det som utvecklar arbetet, så wrappar Scrum med XP på ett väldigt väldigt bra sätt. Dom vinnande paradigmerna som finns i XP fungerar bra ihop.
109	Gustav	Okej, så mer kod-konkret så använder ni mycket XP?
110	Pendleton	Ja, precis, och det är just som jag sa, med parprogrammering, kontinuerlig integration, kontinuerlig test, så fort det finns någonting att testa så kör man.
111	Björn	Så den tekniska biten i XP kompletterar managementbiten i Scrum ganska bra?
112	Pendleton	Ja, det gör den. Det är bra att man är medveten om det när man börjar jobba med metodikerna att Scrum är en projektstyrningsmetodik och XP är en utvecklingsmetodik, om man skall hårddra det, sen så finns det ett visst överlapp med.
113	Gustav	Så det skulle kunna vara så att XP har haft sitt ”bäst-före-datum” och transformerats in lite i Scrum, och Scrum sedan tagit över?
114	Pendleton	Kanske. Man hör ju mer om Scrum nu än XP i alla fall, tycker jag.
115	Björn	Ja, det kände vi också när vi försökte få tag i folk som har arbetat med XP. Nästan alla använder Scrum idag verkar det som.
116	Pendleton	Ja, men Scrum kan inte ersätta XP på det sättet, för det är inte inne på den nivån. Däremot är det många som kör Scrum och använder sig av paradigmerna från XP, för utvecklarnas arbete.

117	Gustav	Vet dom om det, att det kommer från XP tror du?
118	Pendleton	Vi på Softhouse pratar mycket om engineering practicies istället, vi nämner inte XP på det sättet. Vi pratar att det finns good practicies, och läser man rubrikerna så ser man att, det här är nästan Kent Beck rakt av liksom.
119	Gustav	Okej. Man har ju förstått det, det är inga nya grejer han kommer med.
120	Pendleton	Men han tog det till en extrem nivå. Namnet är ju så genialiskt på det sättet. Det är bra att testa, ja, men då gör vi det hela tiden. Vi hymlar inte, utan nu gör vi det här liksom, stenhårt, om det är så bra så ska vi ju göra det hela tiden, så ser vi om det funkar.
121	Gustav	Det är sant. Ja, vi kom ju in lite på den sista frågan…
122	Björn	Ja, hur du upplever XP gentemot andra systemutvecklingsmetoder? Vi var ju inne i början och pratade om vattenfallsmetoden.
123	Pendleton	Alltså, jag gillar agila metoder, annars hade jag inte jobbat på Softhouse. Så visst, tycker jag att XP är bra, och vattenfallsmetoden är väldigt råddigt att jobba med. Eller tja, råddigt och råddigt, det är tråkigt, för man vet att det går fel, från början. Vi gör en jättestor plan, men vi vet att den inte kommer hålla, ja men vi måste göra den i alla fall för det står här i processen, vid möte två skall vi ha planen framme liksom. Ja men, varför det? Alla vet ju att den är fel? Jag blir så störd på sånt, när man hittar sin trygghet i en instruktion på något sätt.
124	Gustav	Något att skylla på.
125	Pendleton	Ja, lite så. Jag har i alla fall följt processen så jag har ryggen fri. Det är mycket mer tänk själv i de agila metoderna, och ansvar på låg nivå, och det gillar jag.
126	Gustav	Är det inte alla som vill vara så här aktiva i sitt arbete?
127	Pendleton	Nej. Det är extremt vanligt, och särskilt på stora företag är det jättemånga som gömmer sig bakom en process eller så. Hittar en trygghet, ja, vi har jobbat så här i tio år, då gör vi det, går till jobbet varje dag och det är tryggt och bra.
128	Gustav	Nu gör vi såhär och sen fikar vi.
129	Pendleton	Men dom har ju lugn och ro på jobbet liksom. För dom går till jobbet, vet vad dom skall göra och sen går dom hem. Ingen kan skylla dom för att dom inte har gjort sitt jobb.
130	Björn	Det kunde ha varit mycket bättre kanske?
131	Pendleton	Man kan ju se det på andra hållet också, fungerar dom bra i en agil värld? Om dom nu tycker att det är så lugnt och dom vill ha det på det sättet, fungerar de i så fall i en situation där de måste ta mer ansvar, eget initiativ och så här, spruta lite mer idéer och vara aktiva? Vill de bestämma över sin dag egentligen, eller vill de ha instruktioner? Det kan jag känna ibland på jobbet också, det är jätteskönt när någon annan bestämmer allting, jag är trött på att bestämma själv. Det är tryggt och behagligt när andra bestämmer, men sen så tycker jag att det blir tråkigt, men det är för att jag är som jag är.
132	Gustav	Men då är det många eldsjälar som kanske lockas till lite mer agila metoder?
133	Pendleton	Ja, det är det ju. Tittar man på Softhouse till exempel så är där mycket kreativa människor, man märker det på våra avdelningsmöten att det är ett himla tjattrande. Särskilt på den avdelningen där jag sitter, med verksamhetsutveckling. Man pratar om att folk är som krokodiler, stor mun och inga öron och det stämmer. Jag var på en sådan avdelning på ett företag i Stockholm i slutet på 90-talet, och där var det värre. Det är lite bättre där jag är nu, folk lyssnar på varandra här inne faktiskt. Men det är mycket åsikter och starka personer, kreativa människor som ska hitta på saker, och läser jättemycket böcker och artiklar och såhär.

134	Gustav	Intensivt ja.
135	Pendleton	Ja.
136	Gustav	Vad tror du om framtiden för det agila arbetssättet, framför allt XP?
137	Pendleton	Det har redan börjat snackas och ryktas löst om ”the next generation”, eller vad man kallar det för. Jag har inte riktigt lyckats tränga in i vad det är mer än att man även kastar de agila reglerna. På något sätt blir det som att man gör en ny metodik för varje projekt, men jag vet inte hur mycket det kommer att slå egentligen. Det som vi väntar på nu och som vi försöker komma in och jobba med är uppskalning; att försöka få det att fungera i stora organisationer, stora projekt. I Scrum-världen så säger Sutherland och Schwaber att de har skalat upp det, men då är de fortfarande i storleksordningen ett par hundra personer. Företag 1 har en agil satsning här i Skåne, och där är det 400 utvecklare.
138	Gustav	Okej, är det deras egna anpassade metod då?
139	Pendleton	Nej, vi har en kille som är inne och hjälper dom litegrann.
140	Gustav	Kent Beck lanserade ju en uppdaterad version, som riktade sig just mot stora företag.
141	Pendleton	Ja, och det har det kommit inom Scrum också. Jag har den boken med mig, som också handlar om uppskalning. Men folk som har provat metodiken i den är väl sådär nöjda.
142	Björn	Det verkar vara samma sak med Kent Becks uppföljare, för betydligt färre refererar eller snackar om den nya versionen, utan det är originalversionen som gäller.
143	Pendleton	Ja, den här är lite mer refererad tror jag, men det är nog för att Scrum är mer använt. Scrum har nog lättare för att slå sig fram därför att projektstyrningsmetodik går in på en högre nivå inom företaget, en sådan utvecklarnära metodik som XP kommer bara in på team-nivå, medan Scrum riktar sig till projektledaren som har rätt så mycket makt, både formell och informell, vilket gör det mycket lättare att få rubriker och få igenom det på ledningsgruppmötena och allting sånt.
144	Gustav	Det är sant. Om några hackers sitter och hittar en jättebra metod…
145	Pendleton	Så är det nästan ingen som bryr sig.
146	Gustav	Nä, det är sant. Om du bara skulle spekulera, vad tror du det är som gör, att XP inte fungerar?
147	Pendleton	Jag har inte tänkt så mycket på det, men det skulle nog kunna ha en viss inverkan det vi nyss pratade om; de kommer inte upp på dagordningen på beslutande möten där man faktiskt bestämmer saker, där man har pengar och makt.
148	Gustav	Skulle metoden fungera om den fick mer uppmärksamhet?
149	Pendleton	Nä, inte i en stor organisation så fungerar den inte ensam, nu har jag inte läst hans nya version över huvud taget, så det har jag ingen aning om, men den gamla versionen i alla fall adresserar inte projektstyrning tillräckligt mycket.
150	Gustav	För dagens behov?
151	Pendleton	Nä precis, det gör den inte, det kan jag säga. Men som sagt, practices för utvecklare är jättebra , verkligen. De flesta av dem fungerar, test first har jag väl inte riktigt fullt genomfört, men lyckas man genomföra det så är det ju kanon, verkligen.
152	Björn	Så XP är bra på det viset att man kan ta vissa tekniker och tillämpa i andra systemutvecklingsmetoder?
153	Pendleton	Ja, det skulle man kanske kunna säga.
154	Gustav	Du hade inte upplevt XP som något stort över huvud taget. Du har sett mycket Scrum men inte stött på XP sedan dess?
155	Pendleton	Precis, inte som att man säger ”nu kör vi XP”, mer att ”nu tar vi det här från XP och applicerar det i vårt Scrum-projekt”.

156	Gustav	Okej. Har XP har varit en del som ledde fram till nya metoder, har det varit viktigt i utvecklingen?
157	Pendleton	Alltså Scrum är inte nyare. Scrum togs fram ungefär samtidigt, nej, XP kanske låg lite före, men nej, det är nog mer att när folk har jobbat med dom så har man sett att de fungerar ihop och att de smälter samman på det sättet.
158	Gustav	Så Scrum har kommit ut lite som vinnare i det här slaget?
159	Pendleton	Ja, just nu. Sen har jag provat en annan inriktning med feature-driven-development, det är en annan agil metod som liknar Scrum rätt så mycket men man jobbar inte med sprintar på det sättet utan man baserar sig på features helt och hållet och paketerar dom, och tar ut en feature i taget, mer asynkront. Scrum skall liksom takta i synkrona sprintar hela tiden.
160	Gustav	Aha, okej. Men här tar man mera detta, nu skall vi kunna detta och fixar det.

B3) Intervju med informant 2

1	Gustav	Jag har ju gått här på datateknik innan, jag är gammal D03:a så att jag kommer ihåg att jag började läsa den här kursen i grupputveckling och då var du en av kunderna, eller skulle vara en av kunderna. Tror det var -05 eller något sådant där.
2	Bendix	Ja.
3	Gustav	Vi har forskat lite och ditt namn har stått med på några papper om XP så att vi är väldigt glada att du ställer upp, ska bli intressant att höra vad du har att säga. Om vi börjar lite med din bakgrund bara kort. När det gäller XP och vad jobbar nu så att säga.
4	Bendix	Ja, jobbar ju inte med direkt med XP jag jobbar med konfigurationshanteringar.
5	Gustav	CM eller?
6	Bendix	CM ja och det är också det som jag gör, jag kör en CM-föreläsning på XP-kursen. CVS-laborationerna. Annars innan jag kom till Lund 2001 hade jag läst lite om XP, agila metoder men det var först när jag kom till Lund som jag började arbete i förbindelse med PVG-kursen. Tillsammans med Görel och Boris så växte det fram.
7	Gustav	Hur populärt var XP och agila metoder vid den tiden, 2001?
8	Bendix	På universitetet var det något man pratade om men inom industrin var det okänt.
9	Gustav	Det var det?
10	Bendix	I stort sätt.
11	Gustav	Så det var opopulärt i forskarkretsar?
12	Bendix	Jag vissa forskarkretsar, speciellt de studerande som uppfattade agila metoder, XP som... vi behöver inte skriva dokumentation, vi behöver vi inte ha disciplin. Kaos, panik, anarki.
13	Gustav	Det känns lite som programmerarnas metod. Och du var inne på när vi pratade med dig senast att det är mycket mer för programmerarna än för ledningen. Det är svårt för ledningen att förstå fördelarna med XP. Är det lättare med t.ex. Scrum att de ska förstå?
14	Bendix	Det är två fördelar med Scrum om man ska försöka införa det i näringslivet. Det ena det är att man kan bli certifierad Scrummaster och kan man bli certifierande så litar ledningen på att, här är en metodik, man har blivit certifierat inom det, och det går säkert bra. Som jag har hört om certifiering det är att man är fysiskt närvarande på en eller tvådagars kurs, så blir man certifierad.
15	Gustav	Som kostar pengar också.
16	Bendix	Som kostar pengar och sånt. Det andra det är att riktigt att Scrum är riktat mot projektledning, du känner till PVG-kursen och projektdelen. Så mycket i Scrum fungerar på The Planning Game och på kunden och säger man i Scrum att sen får teamet organisera sig som de vill internt. De kör den här iterationen eller sprint som det heter.
17	Gustav	Scrum lägger sig inte i huvudtaget hur man skriver själva koden?
18	Bendix	Nej och i XP kan du också i princip göra vad du vill men XP har ju de här praktikerna som säger att om du vill ha hjälp att överleva det här så kör continous integration, frequent releases, refactoring, incremental refactoring, test-first, så på Scrum så säger man: ja nu kör vi en sprint. Så får vi implementera något.

19	Björn	Så Scrum och XP komplementerar varandra väldigt bra. Med management och den tekniska biten. Är det så det ser ut?
20	Bendix	Ja jag skulle vilja säga att om man kör Scrum så ska man titta på XP för teamet så den som är Scrummaster ska se till att teamet fungerar på ett bra sätt och att de arbetar smidigt, så borde de titta mer på XP.
21	Gustav	När du jobbar med CM ser du XP då som en bit i att strukturera eller organisera.. Jo vad tänkte jag på. Jo vi intervjuade Christer Pendleton han sa han kände till dig.
22	Bendix	Jo.
23	Gustav	Han berättade väldigt kort om det, jag kan inte säga att jag förstår exakt vad det är mer än att det försöker skapa struktur i processen. Kopplar det till XP på något sätt? Alltså har de med varandra att göra överhuvudtaget? Konfigurationshantering det är när man har behov att göra i vilket som helst projekt också i agila projekt där XP ingår, Scrum projekt, vattenfallsprojekt men man behöver göra det på lite annorlunda sätt och med lite annorlunda fokus. I Stora projekt är fokus ofta på ändringshantering med change-request. Har ni gått PUSS-kursen? Programvaruutveckling i stora system.
24	Gustav och Björn	Nä.
25	Bendix	Där kör man ett projekt som är mycket vattenfallsaktigt där man har en kravspec och om det ska nya krav in i kravspecen och den är fryst så ska man göra en change-request och så ska det upp på ett möte där man ska rapportera det och det blir mycket trögt och mycket byråkrati men man har behov av det för att ha koll vad som man vill ha ändringar och vad man inte vill ha ändringar och på Scrum så händer det på det man har varannan vecka där man planerar och på PVG där har du ju planning game. Där kunden prioriterar vilka stories han vill ha med och tidsuppskattar och estimerar så lite fokus är det också på det på XP på den biten i ändringshantering. Men det är mycket mera fokus på continous integration, frequent releases, quick builds och sånt. När man har tio, tjugo utvecklare som arbetar med Collective code ownership då behöver man se till att det verkligen fungerar smidigt med att göra commit, merga och kolla att det fungerar och sånt.
26	Gustav	Men borde inte XP göra då att det fungerar bättre då om Scrum inte har några sådana riktlinjer, XP de har ju de där tolv praktikerna. Jag menar jag har själv suttit lite med det, jag läste ju teoribiten och sådär och jag tyckte det verkade väldigt smidigt. Två kan fungera när man sitter och programmerar men det blir problem när man sitter själv utan några regler. XP borde klara av den biten mycket bättre än Scrum.
27	Bendix	Om du är en erfaren programmerar, har en projektledare som har erfarenhet, som vet man vilka saker som går och vilka som går dåligt. Men när man har inte har så mycket kolla eller har något som går trögt utan att man förstår varför, så har man hjälp och stöd från agila praktiker i XP. Så får man lite anvisning, så här bör du göra. Det som Kent Beck har gjort det är att han har skrivit en second edition av sin bok och den är mycket, mycket bättre. Det är en total omskrivning.
28	Björn	När man kikar på det som är mest refererat så verkar det vara så att det är den första utgåvan som refreras mest. Stämmer det eller tycker folk bra om andra utgåvan också.
29	Bendix	Jag är inte intresserad av vad andra tycker. Jag tycker att den andra är bättre. I den första där har han de här tolv praktikerna. I den första där är han väldigt fundamentalistisk. Det är programmeringstaliban så han säger att antingen så följer du alla de här tolv praktikerna och så är du agil eller också så är du inte agil alls. Men typ så är han väl inte taliban som George

		Bush, ”either you’re with us or against us”. Det som han gör i den andra utgåvan är att dels så blir han lite mer filosofisk och ger bakgrund för varför att man gör XP och varför det är bra och vad det är problem som man försöker hantera och vad som är filosofin… dels så har han kommit på lite fler praktiker och några har delats upp i två praktiker så delar han upp de i två grupper. Primary practices och corollary practices och corollary practices är sånt som collective code ownership, customer on site om jag kommer ihåg rätt, det är säkert också frequent releases. Och det är sån som han säger: “Försök inte att göra det här om du inte har på plats primary practices.
30	Gustav	Utan de är beroende av varandra?
31	Bendix	Om du inte har continous integration, så glöm att ha collective code ownership. För det kommer att ge kaos. Parprogrammering det är primary practises. Det kan man göra, man har ha bara parprogrammering och ha bra nytta utav det och var lite extrem, lite agil. Man kan ha test-driven utveckling. Bara göra test-driven utveckling, inte parprogramming, inte collective code-ownership, fortfarande vara lite mer agilt.
32	Björn	Men tycker du att man använder XP då om man bara använder någon praktik? Tänkte på det med skräddarsy, om man anpassar.
33	Bendix	Det är ju en hopplös diskussion. Som väl inte, den bingar inte något konstruktivt med sig. Jag tycker mycket bättre om Kent Becks andra version. Som säger: Här är några praktiker som är användbara i alla kontexter oavsett om du är agil eller använder Scrum eller XP. Här är några praktiker som gör att du kommer att vara mycket agil. Det kommer gå mycket snabbt och smidigt men de här praktikerna, de kräver att du att du ha på plats de fundamentala praktikerna. Har du inte på plats de första praktikerna så försök inte med de andra. Lite som Jackass på tv ”Don’t do this at home”.
34	Gustav	Var han lite naiv i sin första utgåva?
35	Bendix	Nej, jag tror han var…
36	Gustav	Gustav: Han var programmerar verkligen ”det här bra, ta allt detta”. ”Bli extrem med detta”
37	Bendix	Ja, han var kanske för idealitistisk. ”Det är här bra, det fungerar, fantastiskt. Alla ska göra såhär, Man får inte undvika att göra det här för det är också bra.”
38	Gustav	Svårt i praktiken?
39	Bendix	Om du verkligen vill ha det fulla värdet av XP så ska du köra det helt, självklart. Men om man har företag där man säger ”det här med test-first, testdriven utveckling det är bra. Det låter bra, det skulle vi gärna vilja göra. ”Varför skulle man inte göra det då?” Och så använder man den praktiken. Det här med collective code-ownership, varför skulle jag inte kunna göra det? Oups, det var en av de här corollary practices, vad kräver de för att vi ska på ett säkert sätt ska kunna ha collectitve ownership, det kräver continous integration, då måste vi få på plats continous integration.
40	Gustav	I kursen på PVG där har ni fortfarande va de utspringliga 12 praktikerna?
41	Bendix	Ja.
42	Gustav	Det har ni, vad är anledningen till det? Om ni använder den andra utgåvan.
43	Bendix	Det är ju resursfråga. Vi har pratat lite grand om den här second edition när den kom. Så frågade Boris och Görel vad jag tyckte om det och jag sa att den är skitbra men den går inte att använda på PVG-kursen. För de studerande, som inte känner till programvaruutveckling där är den för filosofisk och för flummig.
44	Gustav	Man behöver erfarenhet för att förstå.
45	Bendix	Ja för att värdesätta det som de är. Det här med att man har tolv praktiker, tolv klara praktiker. Det är en liten chromatic bok som man använder nu.

		Den är mycket kort och mycket precis på det här. Det tror jag att det är lagom för studerande, de lyckas att ta in.
46	Gustav	Ja den är bra den boken. Jag tycker den är tydlig.
47	Bendix	Den är mycket tydlig, men den ger inte alla detaljerna.
48	Gustav	Nä det gör den inte.
49	Bendix	När man inte har så speciellt bra koll på det här så behöver man inte alla detaljer. Blir mer ”what ever”.
50	Gustav	Man behöver konkret i början?
51	Bendix	Ja. steg 1 och steg 2 och så läser man det och så lyckas man ungefär förstå, frequent releases, collective code ownership, test-first. Man har ungefär koll på det. När man sen börjar med projektet så upptäcker man inte att man har helt koll på det. Men man lär sig Och tillslut har man hyfsat bra på koll på det här. Också är man redo till att kanske ta en second edition av Kent Beck.
52	Gustav	Men har du sett att det har fungerat bra att använda XP? Du har ju sett att det fungerat bra på programvaruutvecklingen.
53	Bendix	Ja.
54	Gustav	Men har du sett något om något företag som använt det.
55	Bendix	Nej, det är bättre ni pratar med Christian om det. Han har varit runt på olika företag och nu jobbar han som konsult och känner mycket bättre till vad som händer i näringslivet. Men jag tror att det som de kör när de kör agil metod i näringslivet det är Scrum.
56	Björn	Ja det är ju den bilden vi har fått också när vi hört av oss till olika företag, det verkar väldigt, väldigt vanligt med Scrum idag.
57	Bendix	Ja och jag tror att det de två primära sakerna till det är dels att man kan bli certifierad Scrummaster. Certified det låter bättre än att ”jag är agil praktikant, jag är XP-praktikant.” och dels att ledningen som ska besluta om de här och som ska ändra organisationen, som ska ha ut en produkt på marknaden. De vill försäkra sig att det är koll på det här. De är vana vid vattenfallsmetoder och stora processer och sånt. Och de sitter här och så tittar den ner. XP har primärt fokus på utvecklarna och att utvecklarna där arbetar tillsammans och lite grann på releases och planning game och sånt. Men det är inte alltid att de kan se det när de ser ner där och det fattar inte vad som det är och de ser det som många studerade också gör i början ”Ja! Anarki, kaos, vi kan göra vad vi vill, vi behöver inte metodik, vi behöver inte disciplin” Det är så man uppfattar det.” Jag kommer inte ihåg vem det var inom de agila världarna. ”Jag har inte sett något så mycket som kräver så mycket disciplin som agila metoder. Det lär ni er också när ni kör de här PVG-projekten. Att om man inte använder de här metodikerna med contiouns integration, update, update, update, update, commit, frequent releases. Och man inte får till en bra releaseprocess. Så går det åt skogen. Då blir det helt panik. Så man har verkligen behov av mer disciplin på agila projekt än man har på vattenfallsprojekt. På vattenfallsprojekt, där tar man fram sin metodhandbok och så säger man steg 1 gör så här. Dutt, dutt, dutt. Steg två 2 gör så här dutt, dutt, dutt. I agila metoder, Scrum och XP så är det någon riktlinje, ungefär så här ska du göra. Så ska komma ihåg, bra beteende och sånt.
58	Gustav	Tycker du att Beck har missat att tänka på management när han gjorde sin metod. Är han för centrerad kring programmerarna för att få genomslag.
59	Bendix	Nej, XP fungerar bra i vissa kontext. I de kontext speciellt i den kontext där man har en kund som inte helt har koll på vad han vill. Och som är beredd på att vara customer on site och som beredd på att vi så småningom tar fram det här i iterationer men om vi har en kund som har kund på alla krav fram början och som inte är beredd on site. Så fungerar vattenfallsmetodiken lika bra.

60	Gustav	Vad tror du om framtiden för agila metoder?
64	Bendix	Det man hör att nu har industrin upptäckt det här buzz-words. Nu vill alla vara agila. Och så under de nästa 2-3-4 år så kommer alla bli agila. Och så upptäcker man ett nytt buzz-word. Näringslivet upptäcker väl att agila
65	Gustav	Så det var hypat, det var för?
66	Bendix	Nej, men om man inte gör det på det rätta sättet så kommer de inte fungera.
67	Gustav	Okej så du tror att man kommer inte att lyckas göra det som det ska?
68	Bendix	Nej, jag tror att det fortfarande. Detta är risken, det är att det är att det fortfarande är vattenfallstänkande i organisationen och från ledningen.
69	Gustav	Man tar inte till sig det agila tankesättet helt?
70	Bendix	Man litar inte helt på det eller man kan inte släppa tänkandet. Eller att man försöker att köra XP på en projekttyp där XP inte är lämpligt.
71	Gustav	Vi har läst en hel del där de har försökt använda XP och sedan försökt använda vissa praktiker. Man litar inte riktigt på det. Och det verkar också vara mycket diskussion om det att XP ska anpassas. Vissa praktiker fungerar, vissa fungerar inte. Det är svårt att hitta miljöer där det fungerar o.s.v.
72	Bendix	Ja, en annan sak. En sak som är svår med är XP och som gör att XP inte kan fungera i alla kontext, det vi kom fram till på CM kursen just nu och det är att det är skillnad på vattenfallsmodellen på XP. Det är mycket mindre byråkrati och pappersarbete på XP. Så när vi har en change reuqest eller en ny story. Så på XP så görs en ny tidsuppskatning. Kunden estimerar, man kollar om man behöver fråga tiden om det är något som inte är klart så frågar man kunden om något är oklart med storyn. Man har kunden på plats och kan fråga kunden. Man får en story, man gör enhetstest, man skriver test, enhetstest, skriver och sen kör man acceptanstest och sen är storyn implementerad. Det som händer på ett vattenfallsprojekt det är att man får ett change-request, det ska dokumenteras en change-request. Om det inte finns tillräckligt med information, att vi inte kan de exakt vad det innebär så får det gå tillbaka till kunden. Så har vi första skillnaden. Med XP, där har vi kunden plats. Så vi kan omedelbart få feedback och information från kunden. Om det här, vi kör projekt här och SonyEricsson är kunden. Så kan vi ta telefonen, vi kan maila eller faxa och så kan de faxa tillbaka 2-3-4 dagar senare. Ett annat när vi har godkänt change-request så går det till testgrupper som ska göra om de testfall som det ska modifieras, får vi har modifierat de här kraven. När de gjort om testfallen så går det tillbaka till projektledaren, Så säger han ”nytt testfall gjort” så ger han de här till utvecklaren som skriver kod, som gör lite test. Sen ger han det här till utvecklarna som skriver kod, som skriver lite test och sedan går han tillbaka till projektet och skriver test ” nu är det klart” , sen går han till testerna, den här nya koden som ä klart. Testa det med de nya testen ni skrivit. Sen går det tillbaka till projektledaren. Det är mycket, mycket trögt. Och det som fungerar jättebra. I XP det är vem är det som skriver testfallen?
73	Gustav	Programmeraren.
74	Bendix	Vem är det som programmerar?
75	Gustav	Programmeraren.
76	Bendix	Vem ska testa?
77	Björn och Gustav	Programmeraren.
78	Bendix	Han lyckas att koordinera med sig själv förhoppningsvis. Men det som är svårt. Och det som det dels har varit på SonyEricsson det är att 95% av utvecklarna som sitter på SE de vill jag inte lita på att de klarar att skriva bra testfall. De är programmeringsspecialister, inte test-specialister så om

		man har tillgång i ett projekt till mycket erfarna programmerar som kunde skriva testfall och göra allt det här så kan jag vara tester, systemarkitekt osv. Men de som sitter på SonyEricsson några av dem är jättebra på att skriva på att skriva testfall men jättedåliga på att skriva kod. Andra de skriver fantastisk kod men skulle inte kunna testa det här komplett. Viss mån av testning kan de göra.
79	Björn	Så styrkan med XP är den snabba feedbacken och den snabba gången i projektet?
80	Bendix	Ja för att man är agil och för att vi inte ska koordinera och kommuniceras så mycket och att kommunitionsvägen är mycket direkta. Därför customer on site.
81	Gustav	Snabb feedback hela tiden?
82	Bendix	Ja. Och vad kommer att hända om SonyEricsson eller IKEA så också kör med agil metoden. Om de säger, nu kör vi XP. Så säger man praktik ”customer on site!. Yeah right. Det har vi inte resurser till”. Ok vad kommer hända, det kommer hända exakt detsamma som händer PVG-projekten här. Vi har inte resurser till att varje projekt ha en dedikerad kund. När jag är kund så är jag kund för 3 eller 4 projekt. Så jag är inte på plats. Vad kommer hända. Du har en frågat till kunden, var han inte här? Coach, jag skulle gärna vilja fråga kunden om den här. ”Anteckna det så frågar vi kunden när han kommer”. En halvtimme senare kommer kunden. Kanske är du så upptagen med din partner och upptäcker inte ens att kunden har kommit. Eller du upptäcker att kunden har kommit men du har glömt bort att du har någon frågor.
83	Gustav	Har man redan jobbat förbi då?
84	Bendix	Ja, om det är något som du har behov av att få svar på nu, så försöker man gå och leta upp kunden. Annars händer det som inte får hända, nämligen att DU tar beslut för kunden. Du svarar på frågan. Och om industrin verkligen vill köra XP och Scrum och verkligen vill lyckas med det. Så, anser jag, att det är ett måste att de också tar det på allvar. När vi har customer on-site så har vi kommunikation, det blir lätt och smidigt. Det som industrin, näringslivet någon gång.. så fattar man inte att näringslivet de lyckas med att överleva. De tänker inte, när man säger customer on site så säger de ”nej nej nej, det är för kostsamt. Vi kan inte ha en person där åtta timmar om dagen alla dagar”. Men varför inte?
85	Gustav	Köper de inte argumenten med att det levererar mer värde i projektet..?
86	Bendix	”Jo men så mycket mer värde är det nog inte, och han är nåbar på mobiltelefon” Jo men om han sitter i ett möte så är han nog inte nåbar och kommunikation på mobiltelefon, ja så ska man ha numret på kunden. Ja det är för tråkigt. ”Jag han tänker säkert så här så gör vi sådär” Så har vitsen med customer on-site, den är försvunnen.
87	Gustav	Sker det här med andra praktiker också? Att man börjar tumma lite ”Äh, parprogrammering, hittar inge nu… Jag börjar sitta själv”
88	Bendix	Ja det man har upptäckt med t.ex. parprogrammering är att i vissa fall så har vi inte parprogrammering. I vissa saker så har jag inte behov av en partner.
89	Gustav	Är det enkla uppgifter då?
90	Bendix	Ja. Oftast så är det det. I många företag så har man det som heter kodgranskning. Att du skriver någon kod och så har du någon annan som granskar och läser din i kod. Det som händer i parprogrammering det är att kodgranskning den sker parallellt med att du skriver koden. Du skriver koden, parallellt med att du skriver den så granskar jag den. Och är det något konstigt så kan jag fråga dig ”varför gör du det?”, ”Du glömde ett semikolon där”, och om du har behov av ett bollplank eller annan design eller vet inte riktigt hur du ska implementera så har du ett bollplank och

		det är därför som industrin ser det som två personer som gör en persons arbete. Men det är fel för det är två personer som gör två personerna arbete. Parallellt på plats. I Vissa fall så kanske jag får sitta och vänta lite bland på det du ska skriva men i andra fall är man bollplank, man kan ställa fråga, fånga fel innan de bli förstora och växer.
91	Gustav	Så industrin har svårt att förstå XP överhuvudtaget, varför det är viktigt att alla delarna är med.
92	Bendix	Det är mycket ytligt med XP, agila metoder. De kommer inte riktigt på djupet. Customer on-site vi har inte resurser att ha en person där 8h. Om du bara har 2-3 frågor så får han en skrivbord där borta i hörnet och en dator. Han sitta och kolla email, läsa sin email. Han har skriva någon kravspec, han ska göra granskning på något. Han sitter och arbetar det. Med det arbete han har. I princip är det ju också kunden som skriva acceptanstest. Så han har mycket arbete och om ni har behov av att prata med kunden så är han här. Där är han, fem meter, 3 sekunder. Men om jag sitter där borta på rummet. Man har gjort en undersökning om distansen är mer än 25-30 meter så kan ju lika gärna sitta på månen. För det ligger någon psykologisk barriär. På SonyEricsson, det är ett bra exempel, eller Ericsson som har avdelningar runt, på olika ställen i byn. Folk de pratar inte med varandra. De går inte från vattentornet ner på Gastelyckan och pratar med någon. Nej. Uppe på vattentornet. En byggnad och en annan byggnad. Nej. En våning, en annan våning. Nej. Jag kan sitta mig på lunchrummet och så men det är inte så där tät kommunikation och det är orsaken till Kent Beck säger team in one room. Det är smidigt.
93	Gustav	Nu sa innan programmeraren behövde vara väldigt breda i. Kunna testa, programmera, granska, vara tracker osv. Är det för mycket att kräva? Klarar alla att arbeta med XP? Har de den kunskapen?`
94	Bendix	Nej. Det är inte heller alla som klarar av att arbeta med open-source.
95	Gustav	Så det är ytterligare ett problem med att det kanske inte har kommit vidare. Programmerarna isig, det är inte alla som gillar den.
96	Bendix	Det är ju problem om IKEA vill köra agilt och vill köra XP eller Scrum som i princip där har man också har testrollen och utvecklarrollen i sitt softwaredesign… Om IKEA insisterar att göra det med utvecklare som inte är kapabla så… Lycka till. Men om man säger att vi har nya krav, vi har nya förutsättningar, så vi plockar ut ett team av de bästa utvecklarna, som också kan vara testare. Så att vi åtminstone har 2-3 personer inom det här teamet på 15 som är bra på att göra software arcitecture.
97	Gustav	Och så utgår man från det?
98	Bendix	Ja. Så kan man väl tillämpa och anpassa XP så att i extrema fall så ska alla vara bra på att skriva kod och skriva testcases och göra arkitektur och design och sånt. Men vill man inte bara helt extrem men ändå köra agilt så får man se till att det iaf inom varje team gärna finns en person, helst gärna fler än en person, så att det inte alltid är en person som gör testfallen. Men att vi får en grupp där vi har 4-5 som kan göra testcases. 3-4 som kan göra arkitekturen.
99	Gustav	Så man får XP då även om alla inte göra allt? *Lars nickar* Okej.
100	Bendix	Man får fortfarande det även om man har team i one room, så har man en som skriver testkoden och då är det mycket smidigare att snacka och kommunicera och koordinera.
101	Gustav	Ja ni har ju skrivit om det här, men vad tyckte du egentligen. Hur fungerade egentligen att implementera XP bland studenter, jag förstår att ni tyckte det fungerade bra som en lärometod men kändes det som att de som kom ut nu skulle kunna sitta i XP-projekt allihopa och det skulle gå bra? Dom som inte är befläckade av det här vattenfallstänket lika mycket som de som jobbat med det i 20 år.

102	Bendix	Nackdelen som de studerande har efter att de har gjort PVG-projektet det är att de har inte så mycket erfarenhet självklart. De har gått 2-3 år här. Fördelen är att de är inte inbakade i vattenfallstänket. Fördelen är att de har mycket bra erfarenhet med agila praktiker. De har lärt sig the hard way att om vi inte gör update regelbundet och har små tasks som vi kan commita regelbundet så får vi en gigantisk merge-konflikt som det tar fem timmar att reda ut. Det är en fördel om man får en sådan gigantisk mergekonflik för det glömmer man aldrig. Så vet man update, update, update, commit. Vi ska alltid updata. Vi ska ha small tasks. Vi ska inte göra en big bang refactoring. Vi har ju vissa friheter här på universitetet för de studerande de ska lära sig. Och ofta är det som lär sig mest av är misstag. Och vi ser till att det finns tillräckligt med misstag att göra och vi har ju det privilegiet och som jag säger till studenterna när jag kör den här CM-föreläsningen. Så har jag en bild av de här ”headless chickens” kaos och panik och så säger jag att det är så här det kommer att bli under de första iterationerna. Och när jag känner mig lite deppig så tar jag ett varv i datasalen på PVG-kursen och så säger jag till de studerande, och det är deras uppgift, och de lyckas det med, att jag bara har skoj under de första iterationerna. Och det är det vi ser, under de första två iterationerna, första releasen, kaos och panik. Där ingenting fungerar. Men är klart, att fokuserar man bara på de tolv ursprungliga praktikerna och försöker få på plats de tolv praktikerna under åtta timmer under den första iterationen, det lyckas man inte med. Men okej vi fick på plats det med continous integration, vi fick på plats det med att skriva testfall först för det mesta. Och så var det andra saker som vi inte lyckades med men okej då tar vi hand om det sen och när man kommer till iteration5 och iteration 6.
103	Gustav	Då har man alla praktikerna?
104	Bendix	Ja man har hyfsat bra kolla på alla praktikerna, det är inte alla som går lika smidigt, det är någon som fortfarande har problem med releaser, andra har automatiserade releaser, någon har automatiserade releaser redan till release ett. Så är det bara ”release script” 32 sekunder senare så är det en fil klar att köra till kunden. Eller att man provinstallerar det och simulerar det medan kunden ser på.
105	Gustav	Jag kommer ihåg att du berättade att du brukade tajma hur snabbt de fick fram sin release.
106	Bendix	Ja. Några de tajmar det och det är några team som kör konkurrens. Men det är klart att om du har ett releaseskript som automatiskt kan skapa ett rent workspace, som kan göra en checkout av senaste versionen, som kompilerar allt, som kör enhetstesterna, som automatiskt kör alla acceptanstester, som paketerar källkod och application och dokumentation och teknisk dokumentation i en fil och om du gör det på 32 sekunder så betyder att om du kommer ihåg att bara ha fungerade kod i repository.
107	Gustav	Annars så funkar det inte.
108	Bendix	Annars så funkar det inte. Men om du gör det, och har allt det här på plats så betyder det att om du ska göra en release till kunden klockan tre så fem i tre så säger du ”okej.” De andra de har arbetat i panik de senaste timmarna. Men vi håller på stilla och lugnt och implementera, commita.
109	Gustav	Alltid fungerande kod, alltid klar att releasa.
110	Bendix	Okej, fem i tre. Dags att köra releasen. Mmm. Klick. Bop *Lars simulerar med ljud och gester hur en release skapas på datorn *Okej, du har en dator där. Du packar upp det och kollar att vi kom ihåg att skicka med konfigurationsfilerna. Det fungerar bra, mail till kunden. DE (andra) har två timmars produktion på att göra release. Och ni har fem minuter, max.
111	Gustav	Så när det fungerar bra, då fungerar XP väldigt bra? Det är väldigt smidigt och folk trivs med det.

112	Bendix	Ja. Under sju veckor så kör vi ju bara sex iterationer. Och orsaken till att vi bara kör sex iterationer det är att logistiskt sätt så har vi långlabb på måndagar och planeringsmöte på onsdagar och torsdagar. Om vi gör det omvänt så skulle vi kunna få till sju iterationer under sju veckor. Och vi har funderat lite grann på om vi skulle göra det. I början var det logistiska skäl som gjorde att vi bara kunde ha långlabb på måndagar för det krockade med andra kurser. Det tror jag inte det är längre. Men min hållning till det är att köra iteration nummer sju ur ett undervisningsperspektiv, så får vi för lite vinst för de studerandes lärande i förhållande för den extrakostand som det är också för de studerande att köra en extra iteration. En extra iteration det är en 8 h långlabb, 2 timmar programmeringsmöte och 4 h spikes. Så det är 14 timmar. Om vi kan se att de studerande inte får speciellt mycket lärande på fjorton timmars investerad tid, så är det ingen vits i att göra det.
113	Gustav	Okej, det var väldigt intressant. Det var många nya grejer som vi inte har fått reda på innan. Är det något som du känner att du vill tillägga (sagt till Björn)?
114	Björn	Nej.
115	Bendix	Det har vi pratade om tidigare med kunden på plats, jag är kund för tre, fyra team. Vi märker den skillnad det hade varit om kunden hade varit på plats alltid. Än att vara nåbar på email och på mobiltelefon. Det är inte alls samma sak.. Kommunikation. Jag har gjort ett försök en gång där två av mina team satt i datasalen så jag tog med mig min bärbara dator i datasalen och så var jag på plats. Och efter tre timmar gick jag in i den andra datasalen och sa ”jag är i den här datasalen och behöver ni mig så” och de två andra team där var jag fysiskt i samma rum och där blev mycket, mycket mer kommunikation. De kom och frågade kunden och pratade med honom. När jag inte är på plats så blir det, det här med ”okej nu är han inte här. Så gör jag såhär”
116	Gustav	Verkar det som att resultatet blir bättre också när du är där och kommunikationen ökar? Blir det bättre kod då också?
117	Bendix	I vissa fall helt klart ja. Typexemplet det är ju att vi ska ha det här användargränssnittet där man kan mata in fyra nummer och så trycker man ok och så får man tiderna. Då står det i storyn att det ska vara med stora… det ska vara stort för man gör vid marken och det kan vara sol och så. Så får man en prototyp, i vissa fall så får man inte en prototyp, utan de säger att de har gjort det här användargränssnittet. Då säger man ”Okej” och så säger de ”Jag och det är så här stort siffrorna man matar in (Lars visar med fingret ett mått på ungefär 2 cm) ”Nej nej nej, det ska vara så här stort” (Lars visar ett mått på ungefär 8 cm). I vissa fall så lyckas de att göra så stort. I andra fall så måste de göra om designen och layouten helt.
118	Gustav	Och där har man inte kommunicerat.
119	Bendix	Nej, jag var inte där. Det var inte lätt att fråga mig.
120	Gustav	De antog att ”ja det här är nog stort nog” och så satte man det.
121	Bendix	Ja. När man kommer lite längre fram vill man gärna ha en konfigurationsfil, ett team där vi pratade om på planeringsmötet att man behövde en konfigurationsfil där man kunde ställa in starttider och måltider. Och när jag då kom på den tidpunkten när jag sa att ”nu vill jag ha den här storyn klar” så sa det är gränssnittet till editorn för konfigurationsfilen det har vi gjort såhär”. Och då sa jag ”Jag har inte behov av ett gränssnitt till det”. Det är jag som användaren. Det är min mamma som matar tider vid marken, det är riktigt. Men det är jag som tar hand om och producerar resultatlistan och hanterar konfigfilen. Jag är van vid att programmera, jag är van vid att när man specificerar en grammatik för ett programmeringsspråk så kan jag också skriva med en simpel

		texteditor om ni anger att formatet ska vara såhär. Seprareras med komma eller semikolon eller så där. Så gör jag det, det är inga problem. Jag vill inte betala för att ni ska göra ett snyggt användargränssnitt.
122	Gustav	Hade de gjort helt fungerade användargränssnitt alltså?
123	Bendix	Ja!
124	Gustav	Det hade nog tagit lite tid.
125	Bendix	Jag tror det hade tagit fyra, fem timmar. Men så sa jag nej, nej, nej, nej, nej. Det där det får ni ta bort igen.
126	Gustav	Halv iteration där… för några.
127	Bendix	Ja, får ett par. I detta fall så spelade det ingen roll. Men i verkligt fall så skulle jag vilja säga att ”den här storyn. Den har ni uppskattat till 30000:- Nu har ni lagt ner tid för 50000 eller 60000:- plus att ni ska lägga ner extra tid 15000:- på att ta bort det här. Jag betalar 30000:- Så den extra kostnaden får ni själva ta.
128	Gustav	Ja det är intressant. Har du märkt detta på några andra praktiker där man inte använt det och det har blivit problem p.g.a. det?
129	Bendix	Ja alltså contionus integration, test-first. I princip om man inte kör test-first, så kör man test i efterhand. Och det kommer att ge problem. Om du inte kör continous integration så får du merge-konfliker, om du inte använder dig av team in one room. Nu ska vi börja arbeta på den här storyn. Det betyder att vi ska göra om i de här tre klasserna. Okej (Lars reser sig) ”Hallå team! Vi ska nu göra i klass de och de och de är det några andra som håller på med de?” De är två där borta som håller på med den där klassen. Så går vi och pratar med dem. Där borta 3 m 2 sek under. Kort möte där vi frågar dem hur länge håller ni på med de där? ”10 min en kvart” då säger vi ”okej, så gör vi om i de här två först och den här sen.”
130	Gustav	Slipper merge-konflikter.
131	Bendix	Ja, eller vi frågar dem. ”Vad är det som ni håller på med?”, ”Ja vi lägger till några metoder här uppe i början” ja då säger vi ”yes! Vi behöver metoder så lägger vi till dem i botten” vilket gör att vi slipper merge-konflikter. Kommunikation och feedback.
131	Gustav	Har du märkt någon del av XP som du känner spelar inte så stor roll, den hade kunnat tas bort?
132	Bendix	Nej, jag tror det beror mycket på vilken projekttyp och vilken typ av utvecklare det är i projektet. Där får man nog säga att det agila tänkandet, som säger att det vi har behov av, det gör vi. Det som vi inte behöver det gör vi inte. Finns det ett behov av att skriva dokumentation så skriver vi dokumentation. Om man skriver dokumentation och upptäcker att det är ingen som läser dokumentationen så säger vi ”okej, då är det inge vits med att göra det”. Men om vi är ett konsultbolag som tar fram en applikation för SonyEricsson. Som SE använder och gör underhåll på. Om jag var projektledare på SE så skulle jag uppskatta mycket om det var dokumentation skrivet. Om det är samma team på Softhouse eller var det är som kör underhåll med samma personer så har man inget behov av dokumentation för då kan jag gå och fråga dig. SÅ kan jag göra en väldigt lättviktigt dokumentation. Så om jag har fråga till story nr. tretton om hur den blev implementerad. Det var något här och jag ser att det är du som har implementerat den så har jag inte behov av att du har skrivit dokumentationen om du är här på plats så om jag har namnet så går jag och letar upp dig.
133	Björn	Så det är beroende av vilket projekt och vilka medarbetare om man kan lägga till eller undvika några praktiker eller tekniker?
134	Bendix	Det är svårt att säga.
135	Gustav	Kan erfarenhet kompensera. Att om man har erfarenhet så kan man kompensera på annat sätt.

136	Bendix	Det är klart att om det är erfarna programmerare så har det förhoppningsvis en bra koll på vad som fungerar bra, vad som fungerar dåligt och om de försöker komma lite djupare ner på det agila tänkandet så har de också möjlighet att tillämpa och säga ”okej, XP i praktiken det föreskriver det här. Men i det här fallet kan vi skriva lite mer slarvigt. Vi uppskattar inte parprogrammering. Om man inte kan se någon vits av det så.. (Lars rycker på axlarna) I ett företag där du kör kodgranskning så man ju nästan i den situationen att två personer gör en persons arbete. Så det kan vara en orsak till att man kör det lite då och då. Eller så säger man ”vi kör inte parprogrammering” men vi kan fortfarande använda det här med bollplank, om man sitter i samma rum så kan man säga. Har ni tid 5 min och bollplanka lite den här designen med mig? Ja så går vi över och gör lite bollplank och spånar lite på det här. Det var bra, tack för det. Och så går man tillbaka och är mer produktiv.
137	Gustav	Har du någon annan sådan åsikt om XP som du vill dela med dig? Som inte har kommit fram.
138	Bendix	Nej, konklusionen den är att jag uppskattar XP mycket, jag tycker XP är jättebra. Men XP är inte svaret på alla frågor. Så om jag skulle ta fram kontrollmjukvara till säg ett kärvkraftverk, om det är Barsebäck så skulle jag inte vilja använd XP (Lars Bendix är dansk), om det är Ringhals eller så långt bort så kvittar det väl.
139	Gustav	Det är inte nära Danmark, det blir finlands problem.
140	Bendix	Om jag har duktiga folk i mitt team, lyckas sätta ihop ett team som man i Scrum säger är crossfunctional så att man lyckas ta hand om allt alla uppgifter innehåller då skulle jag inte ha något problem med att köra XP eller Scrum. Om jag inte lyckas med att få till det så är det roller och uppgifter som inte blir täckta. Och så har vi ett problem.
141	Bendix	Följande saker sades av Lars efter att intervjuen formellt var avslutad. • **XP fungerar bra med duktig och kunniga personer.** • **Vattenfallsmetoden är inte alls vansinnig.** • **Metoder utvecklas inom akademin.**

B4) Intervju med informant 3

1	Björn	Vi kan börja med en enkel fråga, vad gör du här på företaget och vad har du gjort innan?
2	Lundgren	Jag har jobbat här på Create i sex år ungefär. Jag är konsult, det är ett konsultbolag. Även teamchef över våra javakonsulter. Jobbar med diverse kunder, jobbar just nu med ett företag i Malmö. Tidigare har jag jobbat på en statlig myndighet under ganska lång tid. Andra stora kunder i regionen som IKEA och Orange. Och innan dess jobbade jag på framfab också.
3	Björn	Vad har du för utbildning?
4	Lundgren	Jag har en ett och ett halvt-årig utbildning efter gymnasiet.
5	Gustav	Tekniskt inriktad då eller?
6	Lundgren	Ja precis.
7	Gustav	Okej. Du har sagt att du har jobbat lite med XP, vill du berätta lite om dina erfarenheter?
8	Lundgren	Mm, nu kommer jag säkerligen blanda lite här, jag har inte kört bara XP utan det kommer gå ihop lite med Scrum. Jag har arbetat med ett antal olika projekt och kan börja med det som jag jobbar med just nu. Där är vi i en situation att vi har en ny kund som egentligen behöver förnya sin utvecklingsavdelning. De kommer att byta teknik och de kommer även att titta på hur de arbetar rent praktiskt. Vad vi håller på med just nu är att vi implementerar en sorts agil metod som baseras på ganska små iterationer, eller sprintar och parprogrammering då eftersom en del av utvecklarna är ganska nya på plattformen. För att de ska komma upp i lite speed så har vi valt att dela upp det så.
9	Gustav	Hur mycket om XP känner du till rent teoretisk? Är det de här tolv praktikerna som du har läst om eller har du gått djupare än?
10	Lundgren	Nu har ju jag jobbet med det i tidigare projekt också så att jag kan väl inte påstå att jag har använt alla tolv praktikerna men en del. De som jag absolut tycker är mest vettiga det är ju… Parprogrammering är ganska bra, jag förespråkar kanske inte att man kör parprogrammering alltid, det finns ganska intressanta konstellationer också när det gäller parprogrammering men det kan jag återkomma till senare. Stand-up meetings är jättebra det också. Det är också en del i Scrum, de är ytterst lika. Test-driven development givetvis, det är väl de första jag brukar se till och få in och det levererar en vettig kodkvalitet.
11	Gustav	Kör ni med det här med continous integration och så…?
12	Lundgren	Absolut. Det blir ett rätt stort paket.
13	Gustav	Ja det blir det va, det hänger ihop.
14	Lundgren	Mm.
15	Gustav	Vet du några delar som inte har varit så viktigt eller som inte har spelat så stor roll?
16	Lundgren	Nej... Det jag kan säga är att man inte ska vara helt renlärd efter vad som skrivs. Man plockar det som känns vettigt, men det gäller i försig vilken metod du än använder.
17	Gustav	Men du har några praktiker som du tycker är fördelaktiga i alla fall.
18	Lundgren	Mm.
19	Gustav	Blir det så att man automatiskt skapar sin egen uppfattning om vad som fungerar?

20	Lundgren	Min roll när jag kommer inte i nya organisationer hos kunder, de senaste gångerna så har de kommit från något annat tex. Vattenfall, RUP eller något annat. Beroende på förutsättningarna hur det ser ut i teamet. Man får gå på erfarenheten och välja det som man känner passar t.ex. parprogrammering om man har olika nivåer på utvecklingsteamet.
21	Gustav	Ställer det mycket krav på utvecklarna? En sån här metodik?
22	Lundgren	Ja det tycker jag. Ja och Nej. Det ställer nog rätt stora krav på hur teamet är i helhet. Du kommer aldrig att lyckas implementera det här i ett team med väldigt juniora utvecklare. Samtidigt tror jag att du får svårt att implementera det i ett team där du har väldigt seniora utvecklare.
23	Gustav	Är de rigida då?
24	Lundgren	Nej det är snarare så att tar de dem som är riktigt seniora så är det oftast dem som har en uppfattning om att så har vi alltid gjort.
25	Björn	Tröga i förändring?
26	Lundgren	Exakt. Så jag tror att en mix är ganska bra och det passar rätt bra in i hur verkligheten ser ut också. Det är lätt att titta i en bok och säga ”Vi har tjugo utvecklare som har tio års erfarenhet.” Verkligheten är väldigt sällan så.
27	Gustav	Vi får fått lite den uppfattningen att det ställer en hel del krav på utvecklarna, de har ju de här delarna att man ska testa, vara testare, man ska programmera, designen i någon mån osv. Många roller helt enkelt. Man ska vara väldigt mångfacetterad. Är det svårt i praktiken att hitta folk som klarar av alla de här delarna?
28	Lundgren	Ja. Det tycker jag absolut. I de fallen jag har jobbat så har det varit väldigt svårt, ibland har man hittat några personer i teamet som kan ta allting. Oftast blir det gärna så att någon är lite mer specialiserad, någon är kanske lite bättre på design eller på vad det nu kan tänkas vara. Testbiten är något som jag tycker alla ska ha bra koll på. Det är snarare ett krav.
29	Gustav	Ja det har med kvalitet att göra.
30	Lundgren	Precis.
31	Gustav	Organisationen då, ni kör XP här in-house.
32	Lundgren	Nu kör vi inte så mycket inhouse, det är oftast ute hos kunder.
33	Gustav	Okej, oftast ute hos kunder. Vad ställer det för krav på organisationen? Måste det vara tillåtande, högt i tak…
34	Lundgren	De flesta implementationer som jag har sett av både Scrum och XP är väl egentligen… Det är bottom-up. Det kommer aldrig ovanifrån. Det är först de senaste året ens som man börjar se kunderna komma med någon form av agila metoder. Då är det oftast en Scrum-variant. Deras anledning att införa det skiljer sig ganska ofta mycket från utvecklingsteamets perspektiv.
35	Gustav	Så för kunderna är det mer en sorts kvalitetsstämpel eller marknadsgrej, buzz-word eller något i den stilen.
36	Lundgren	Buzzword och den störta vinsten brukar vara i att man får bättre precision i vad du kommer att få levererat i de olika punkterna.
37	Gustav	Så det är en utveckling mot att kunderna förstår mer om hur utvecklingen egentligen går till?
38	Lundgren	Det tycker jag nog att man kan se att förståelsen ökar lite. Vilken egentligen bara gör det enklare att införa sånt här. Finns det medvetenhet om varför man måste ha enhetstester till exempel så blir det betydligt enklare. Den absolut svåraste punkten som jag har sett att sälja in är absolut parprogrammering. Det är enorm svårt att sälja in. Oftast är den man försöker sälja in det till någon form av management, någon som sitter med budgeten. Varför ska vi ta två utvecklare och göra någonting när vi kan ta två utvecklare och göra dubbelt så mycket jobb. Så att köra ren parprogrammering är väldigt svårt att införa. Däremot är det mycket lättare

		att få in den som en inlärningsbit när man ska ta in nya personer i projektet eller öka kompetensen på de som redan finns. G
39	Gustav	Man får ta till lite alternativa metoder för att förklara varför det är viktigt?
40	Lundgren	Exakt.
41	Gustav	Är det några andra sådana här praktiker, jag tänker till exempel på ”Customer on site”, är det något ni tillämpar förresten?
42	Lundgren	Nej.
43	Gustav	Har du testat det någongång?
44	Lundgren	Nej.
45	Gustav	Du har ingen åsikt om det egentligen?
46	Lundgren	Nej.
47	Gustav	Okej, nä det har annars varit en praktik som vi har förstått har varit väldigt svår att sälja in. Att alltid kunden ska vara på plats och kosta pengar och så där. Men tycker du att XP fungerar bra överhuvudtaget?
48	Lundgren	Det tycker jag det gör. Som jag sa innan också så har jag inte kört XP rent, det är en bra match med andra metoder till exempel Scrum. Som praktik själv så saknar den ganska mycket kan jag tycka. Den är väldig utvecklingscentrerad som process. Bara köra den där, då kommer vi fortfarande att behöva någonting som kan hantera längre upp i kedjan så att säga.
49	Björn	Så den är väldigt utvecklarnära och inte så mycket management?
50	Lundgren	Ja precis. Det är där ofta konflikterna har varit innan, hur man implementerar sakerna. Projektledare och beställer de har ofta andra perspektiv på hur projektet skall göras, det ska vara billigt och det ska vara…
51	Gustav	Det finns ett glapp där mellan XP och verkligheten nästan?
52	Lundgren	Absolut.
53	Gustav	Så det är en brist hos XP?
54	Lundgren	Nej jag tycker inte det är en brist i XP, det finns jättemånga sätt att bygga ihop något eget och få hela kedjan. Så kompletterar man då Scrum med något annat så får man ganska komplett.
55	Gustav	Så XP är egentligen bra alltså det fungerar när man använder det med men som sagt det är inte helt komplett.
56	Lundgren	Exakt.
57	Gustav	Men när passar det inte att använda XP?
58	Lundgren	XP passar nog betydligt bättre om man börjar från början i något nytt projekt eller någonting. Implementera i efterhand är betydligt svårare. Säg att du har ett projekt som har rullat 1-2 år eller kanske ännu längre. Så är det ganska svårt att gå in och implementera XP i det. Det finns många olika aspekter på det. De har kanske inte en test-suite, de har kanske inte alltid annat runt omkring. Men för nystartade projekt eller som precis kommit igång så är det bra.
59	Gustav	Tänkte mer på det här med anpassning och krav på utvecklarna egentligen. Hur brukar deras syn på agila metoder vara? Framförallt XP. Är det något som man ser som väldigt strikt, väldigt svårt att ta in eller är det liksom härligt för det ”passar oss”?
60	Lundgren	Både och. Det brukar vara ganska övervältigande till att börja med. Framförallt testningen brukar det finnas lite motstånd i början för nyttan med det. ”Varför ska jag skriva test? Det tar lika lång tid som att skriva koden.” Men så fort de har gjort sin första refactoring som har sönder kravbasen så förstår de liksom nyttan med det. Annars tror jag att metod som XP är ganska omtyckt av de flesta utvecklarna om vi räknar bort de som ”såhär har vi aldrig gjort förr”.

61	Gustav	Nä vi har ju forskat en del vi har ju studerat den här väldigt mycket. Har du sett? Läst den?
62	Lundgren	Mm. Fast min är nog en äldre edition tror jag. Den är grön. Grön på sidan. Jag tror den är grön.
63	Gustav	Aa, det finns en grön som är andra utgåvan. Har du med den?
64	Björn	När jag har med en annan men den handlar om planering så det är nog inte den heller.
65	Gustav	Det här var väll egentligen hans ursprungsversion då *pekar på boken* där han förklarar allting och det kan ju verka som att på ett sätt är det ganska enkelt, det är tolv praktiker. Men han ju en otrolig mängd information som underbygger hans filosofi, hans värderingar och så som han vill befästa i de här praktikerna. Och det känner vi också att dels verkar han vara lite idealist, att alla ska det till det extrema. Är det något du känner igen?
66	Lundgren	Absolut.
67	Gustav	Och dels så känner vi också att han tar sig lite vatten över huvudet för det är lite för mycket för en vanlig programmera att ta in allt det här. Han verkar väldigt skicklig men det känns som att det behöver vara lite enklare för att kunna tillämpas fullt ut så som det är tänkt.
68	Björn	Ja det verkar nästan så också men det är kanske för att XP inte har alla praktikerna, att ni anpassar metoden, antningen med RUP eller SCRUM.
69	Lundgren	Jag tror att ett säkert sätt att misslyckas med att införa XP det är försöka införa allt. Man kanske kan införa allt men inte på en och samma gång. Då får du väldigt mycket overhead på varje sak du ska göra och då blir det annat du ska göra och du får motstånd. Handledaren i teamet kommer att tycka att det blir jobbigt. De tycker det är jättebra men så fort det börjar bli ivägen för det vardagliga arbetet så tappar de sitt värde. Så i slutändan så är du utvecklare som ska leverera kod. Så är det. Det är inget självändamål att följa…
70	Gustav	Vad tror du är framtiden när det gäller agila metoder och systemutveckling. Om vi säger tre år.
71	Lundgren	Det är på stark frammarsch. Det är egentligen inget nytt, det har funnits ett bra tag. Men det har vunnit väldigt mycket mark de senaste åren. Om man tittar på de större kunderna. De andra metoderna brukar kanske testas i lite mindre skala först, kanske mindre produktbolag som testar det först. Den typen av saker. Nä jag tror absolut att det är ett bra sätt och jag tror att det kommer att växa. Sen i vilken form om det är XP eller vad det vet jag inte men tankesättet ligger helt rätt.
72	Gustav	Och ni har sett att det har fungerat och gett bra resultat också.
73	Lundgren	Precis. Det är ju så att man måste köra de här lite innan man kan säga att det fungerar också och det är väl det läget man har kommit till precis nu.
74	Gustav	Vi har förstått att Scrum är ju verkligen det här buzz-wordet. Det är bara att läsa Computer Sweden så står det någonting om det varje utgåva. XP har inte alls fått det här genomslaget?
75	Lundgren	Nej.
76	Gustav	När vi kollar igenom artiklar så får vi en känsla av att storhetstiden, eller storhetstiden, 2002-2003. Det är då det skrev mest om det i alla fall. Jag vet inte om det har kommit i skymundan av Scrum men…
77	Björn	Sen kan det ju vara så att arbetsmarknaden ligger lite efterhand i jämförelse med det akademiska.
78	Gustav	Ja, det är sant. Så vi undrar lite, kommer XP i skymundan av Scrum? För vi har sett lite om att man använder det som ni gör, man kompletterar de här två.
79	Lundgren	Jag vill väl nog säga att det är nog inte konkurrenter, de är ganska mycket överlappning mellan de olika metoderna så att ersätta XP tror jag inte att det kommer att göra. Det kommer nog att gå ihop, de har lite olika funktioner.

		Scrum kan man väl säga hanterar egentligen projektet i större skala kan man väl säga. Scrum är mer hur projektet körs XP är hur man gör det praktiska arbetet. Scrum säger ju egentligen inget om hur man gör i en sprint. Om du parprogrammerar eller skriver tester, det är inget som den dikterar. Men de kompletterar varandra ganska bra.
80	Gustav	Har du arbetat med Scrum men inte med XP? Eller dem kommer automatiskt in de teknikerna.
81	Lundgren	Det har nog varit en kombination alltid. Det sker omedvetet. Det är lite som en verktygslåda, man får ta det verktyg som man behöver.
82	Björn	De projekt du har varit delaktig i eller har haft kontrollen över, hur många utvecklare har varit med?
83	Lundgren	Det projektet jag sitter i nu är åtta utvecklare ungefär. Ganska seniora allihop. Projektet innan som jag satt i i tre år var ett betydligt större projekt så hade vi arton utvecklare som mest. Två Scrum-team körde vi. Det är ganska likt de projektet som jag kör nu också. Det är egentligen… Alla utvecklare som var involverade i det var ganska seniora men det var en helt ny plattform för dem. Så vad vi gjorde där egentligen var att vi tog in erfaret folk som kunde träna upp de andra genom parprogrammering och göra en initial design sen fick de fasas ut efter hand. Det projektet var egentligen… Det var en statlig myndighet som skulle byta teknisk plattform för hela myndigheten. De var en utvecklingsavdelning på hundra personer ungefär och detta var första projektet ut så vi handplockade in tolv ifrån resten av organisationen som var de som var erkänt duktiga på att göra saker. Mot slutet av projektet och när projektet var slut så putsades de ut till andra projekt som skulle köras då.
84	Gustav	Det var ett viktigt projekt då?
85	Lundgren	Det var viktigt ja. Och det gick bra.
86	Gustav	Organisationen, vi var inne lite på det att sälja in på kunden. Med Scrum så kan du ju vara certifierad scrummaster t.ex. Spelar det någon roll när det gäller att sälja in en metod? XP kan du ju inte certifiera dig i vad vi vet.
87	Lundgren	Scrumcertifieringen, jag har själv en sådan och kan ju påstå att den är rent kunskapsmässigt noll värd. Du kan sova… Du går en kurs på två dagar och du kan sitta och sova på föreläsningarna. Det är något att ha på papper. Till viss del så kan det kanske påverka. Försöker du sälja det högre upp i hierarkin så är kanske certifiering någonting som är bra. Men jag tror återigen att vi är inne på att det fyller lite olika luckor. Scrum och XP. Ska du köra Scrum så är det vanliga att du tar systemarkitekten eller projektledaren som bli Scrummaster. Så du får mer businesstänk in i det, mer budgetmässigt. Uppåt.
88	Gustav	Som om jag förstår dig rätt så när XP aldrig högre än kanske projektledaren.
89	Lundgren	Nej, inte vad jag har sett.
90	Gustav	Det är han som säger ”Scrum säger att vi ska implementera det här och de här teknikerna kör vi” sen kan han i för sig få problem med att sälja in vissa av de här teknikerna då. Men då går i alla fall inte högre så att säga.
91	Lundgren	Jag tror inte och jag tycker inte att XP… Det är oftast något man gör och inget man behöver sälja in. Det funkar mer om hur teamet fungerar. Men det är klart har man tio par så får man ha med i beräkningarna att man ska skriva tester till exempel. Eller så kan man skita i att göra det för man vet att man inte kommer att få bättre kvalitet på koden i slutändan så man räknar ihop. Parprogrammering är ju typiskt men det är ju precis det problemet vi pratade om innan. Sälja in det är oftast ganska svårt.
92	Gustav	Det är intressant för egentligen är ingen av de här praktikerna nya. Vi känner lite att XP är ett samlingsnamn för de här praktikerna och sedan tar man lite vad man behöver så att XP fyller en upplysande roll snarare än att vara ett regelverk. Är det din uppfattning också?

93	Lundgren	Absolut. Min reflektion också när jag läste boken för en massa år sedan var just det att han ville att man skulle köra hela kittet, att du ska köra väldigt renlärt och det fungerar inte. Välj de praktiker som fyller funktion.
94	Gustav	Känner du att du har något mer?
95	Björn	Nej, inget relevant.
96	Lundgren	Jag kan ju berätta jag gjorde ett intressant experiment i det där stora projektet med just parprogrammering. Vi hade ett antal testare egentligen som annars normalt sätt annars testar applikationer alltså inga utvecklare. Vad vi gjorde där var att vi tog testar och satte de att parprogrammera med utvecklarna. Anledningen till att vi gjorde det från första början var att de hade ganska lite att göra just för tillfälligt. Det var väldigt intressant att se statistiken på kodkvaliten och se de här paren jobba. För deras tester som de gjorde var väldigt olika. Utvecklarna hade ett helt annat perspektiv när de skrev tester än testaren som satt vid sidan om. Så det var faktiskt ett väldigt intressant perspektiv att applicera det på också så det behöver inte vara två utvecklare.
97	Gustav	Hur blev resultatet då egentligen? Blev det bättre kod?
98	Lundgren	Ja det… Alltså det är svårt att mäta. Men jag tycker det för att testaren hittade betydligt mer, eller fler tester att köra på kodbasen än vad bara utvecklarna hade gjort.
99	Gustav	Så det är återigen kraven på utvecklarna att vara holistiska, allsmäktiga.
100	Lundgren	Ja, alltså en utvecklare täcker in det mesta ofta men jag tror snarare det har att göra med tankesättet. Är det en van testare som testar applikationer eller du nu testar så får du som utvecklare en känsla rent tekniskt var i koden det kommer att gå sönder. Så får en testare samma känsla att matar jag in negativa värden i min miniräknare så går det sönder. Vad de tnu kan vara. Så det kompletterar varandra ganska bra. Det är inget man ska göra alltid men har man en kritisk komponent som måste vara vältestat eller så har man problem med en komponent i efterhand så är det bra att göra så.
101	Gustav	Så i praktiken så kompletterar man varandra i teamet då?
102	Lundgren	Precis och just med parning så tycker inte jag att det behöver vara två utvecklare heller. Ett exempel är att vi har parprogrammerat också med en DBA alltså en databasperson, som skriver databasaccess. Han vet exakt hur saker och ting ska vara för att det ska bli bra performance i det hela.
103	Gustav	Då kan han vara med från början där?
104	Lundgren	Precis. Så jag tror synen att en person ska kunna allt… Jättebra på pappret men det finns inga sådana människor eller det finns väldigt få av dem.
105	Gustav	Åter igen, han är idealist?
106	Lundgren	Mm.
107	Gustav	Är det något mer som du känner att du vill ta upp, berätta eller få med?
108	Lundgren	Jag vill bara slå ett slag också för det dagliga mötet, daily scrum eller stand-up meeting. Får man det till att fungera så kan man spara ganska mycket tid också.
109	Gustav	Vad tycker du om XP avslutningsvis?
110	Lundgren	Jag tycker det är en bra metodik, det gäller att plocka de grejer man behöver använda på just det projektet. Man kan inte säga att de här fyra praktikerna behöver jag använda i alla projekt utan det kan vara att det kan skilja mellan projekten också.

B5) Intervju med informant 4

1	Björn	Är det okej att vi anger vem vi har intervjuat eller vill du vara anonym?
2	Lundström	Ni får gärna ange vem ni har intervjuat.
3	Gustav	Berätta lite om din bakgrund, både generellt och när det gäller XP.
4	Lundström	Jag har pluggat civilingenjör i industriell ekonomi i Linköping, blev klar 2002-2003 ungefär, vid årsskiftet där, lite beroende på hur man räknar. Jag upptäckte att det var skoj och intressant med programmering när jag pluggade där, programmera och bygga system och så där. Från och med då bestämde jag mig för att gå in i IT-svängen, hålla det rätt så kodnära också.
5	Gustav	Precis efter bubblan där nästan?
6	Lundström	Precis, man kan säga att jag kom ut när det inte var någon bubbla längre, det var en liten våt fläck på marken, så det var lite tungt. Sen början på 2004 har jag jobbat med IT kan man säga, framför allt med systemutveckling i lite olika roller. Projektledning, lite testledning och en hel del utveckling. Det är väl så att jag har kommit in på XP igenom testdriven utveckling, att jag har insett att det är jävligt bra att skriva tester, och sen så kollar man vilka som tycker att det är bra att skriva tester, jo, det är de som håller på med XP, vad tycker de? Jaha, de tycker såhär och såhär, och ungefär parallellt med XP så har jag snöat in på Scrum. Jag försöker väl i de projekt jag befinner mig i hålla de så agila som möjligt, så mycket av XP:s grundläggande värderingar, alltså testdriven utveckling, engineering practices, källkodshantering och kontinuerlig integration, samtidigt som man kör projekt genom Scrum och iterationer, som XP också specificerar.
7	Björn	Hur länge har du jobbat på SoftHouse?
8	Lundström	Sen i påskas, ett drygt halvår alltså. Sen i slutet av mars.
9	Björn	Hur mycket teori känner du till om XP, mer än de tolv praktikerna? Känner du till mer om de bakomliggande tankarna?
10	Lundström	Nej, egentligen inte. Jag har inte läst speciellt många böcker om XP.
11	Björn	Har du stött på Kent Beck där han uttrycker metoden för första gången?
12	Lundström	Nej. Vad heter den boken?
13	Gustav	Extreme Programming explained.
14	Björn	Skriven 1999 har jag för mig.
15	Lundström	Man läser så mycket man hinner och det är vissa saker man inte hinner läsa liksom.
16	Gustav	Vad vet du för teori egentligen, är det de här tolv praktikerna du känner till då?
17	Lundström	Ja, det kan man väl säga. Det ligger på den nivån ungefär. Jag kan inte säga att jag har en teoretiskt underbyggd bas om varför man ska göra ditten eller datten, utan det är mer så att det känns som att de projekt jag sitter i fungerar mycket bättre om man kör testdriven utveckling eller har kontinuerlig integration eller har en schysst källkodshantering.
18	Gustav	Hur väljer du vilka praktiker du ska använda, eller använder du allihopa?
19	Lundström	Jag kollade faktiskt igenom den listan nu inför det här, och då upptäckte jag att jag använde nästan allihopa, men inte på grund av något aktivt val. Jag väljer inte XP för att jag ska köra XP, utan jag har på något sätt tittat på praktik för praktik, och sagt okej, testdriven utveckling, det vill jag använda, och den där delen, vill jag använda., den där delen, den vill jag använda. Det jag använder minst skulle jag tro är parprogrammering. Där är vissa, vad ska man säga, XP-talibaner, eller väldigt hårda på vad XP är, säger i princip att

		man inte ska skriva en rad produktionskod utan att sitta två, och den ändå gången du kanske kan få sitta en är om du ska skriva lite fler tester. Så gör vi inte i projektet som jag befinner mig i nu. Det är ett rätt så litet projekt, det är fyra personer allt som allt.
20	Gustav	Har ni ”customer-on-site”?
21	Lundström	Ja, enligt Scrum så har vi en produktägare, som är någonstans emellan produktägare och vad Scum kallar det proxy-produktägare, där vi har en person som har väldigt mycket domänkunskap. Sen har vi två personer som är själva kravställarna. Hans jobb är att samla i /solla/filtrera/merga de här två olika vyerna på systemet som de två personerna har.
22	Gustav	Har utvecklarna möjlighet att ställa frågor direkt kravställarna?
23	Lundström	Ja.
24	Gustav	Men de är inte i samma rum så att säga?
25	Lundström	De är inte i samma rum, men däremot så är vår produktägare i samma rum.
26	Gustav	Så de har ändå någon de kan fråga?
27	Lundström	Ja, precis.
28	Gustav	Om vi ändå är inne lite på de här olika delarna, är det några delar som du känner fungerar bättre än andra?
29	Lundström	Ja, det skulle man väl säga. Det finns vissa saker som jag aldrig skulle kunna leva utan, eller som jag känner att, okej, det här funkar inte om ni tar ifrån mig det här, och det är framför allt en schysst configuration management, och sen en SCM-miljö, typ Subversion eller vad fan som helst, plus en kontinuerlig integration, det måste finnas. Sen finns det andra saker, customer-on-site är bra, klart bra att ha, men finns det inte så får man försöka cope with it. Testdriven utveckling är bra ofta, inte kanske alltid. Det finns vissa gånger man kan tänka sig att skippa det, om man ska göra någonting som är väldigt väldigt litet eller oviktig. Alltså, SCM och en CI-hantering måste finnas och har man inte en byggserver så har man inget projekt tycker jag.
30	Gustav	Hur kommer testning in i bilden, är det en grej som du ser som väldigt viktigt?
31	Lundström	Ja. Jag personligen tycker att det är väldigt viktigt. Däremot så finns det en hel del personer som tycker att testdriven development enligt talibanspecifikationer, alltså att skriva en rad kod och sen skriva test, som Beck gör i sina teoriböcker.
32	Gustav	Det här test-first?
33	Lundström	Ja, precis. Det behöver inte finnas där tycker jag. Det kan snöa till det liksom. Däremot så tycker jag att det är väldigt viktigt att man har en schysst coverage, dels på unit-tester och dels på integrationsnivån.
34	Gustav	Så det är flera praktiker som du tycker är bra att använda, men du använder de inte, vad ska man säga, taliban-aktigt?
35	Lundström	Precis, inte slaviskt.
36	Gustav	Och det är för att anpassa det till situationen då?
37	Lundström	Ja, så är det väl. Det känns som att man är lite väl inproduktiv om man fastnar i den här lilla TDD-strömmen, skriva en rad test, kompilera, nej det gick inte, så skriver jag klassen jag ska testa. Man är lite konstruerat inproduktiv tycker jag. Sen så har TDD-gänget , de som följer TDD slaviskt, de argumenterar för att med TDD så bygger du upp en arkitektur och en design, vilket jag inte riktigt håller med om faktiskt. Man bör ha en grunddesign utstakad redan innan och en grovarkitektur redan innan, det kommer liksom inte per automatik av någon magisk ängel som lyckas göra så att man skriver koden på ett bra sätt. Man måste tänka lite up-front.
38	Gustav	Om vi vänder på frågan, är det några praktiker som du känner fungerar sämre?
39	Lundström	Jag tycker att parprogrammering funkar bra i vissa fall. Jag brukar framför allt

		använda det när man ska arbeta ut någonting när man inte har någonting specat, vi behöver liksom arbeta upp en design eller sprida kunskap från en person till en annan, då funkar parprogrammering bra. Det tar lite väl mycket tid att sitta två personer på samma dator jämt. Man får ta det när det behövs, då har man en parprogrammerings-session på 30 minuter och sen så går man tillbaks och återgår till sitt skrivbord igen.
40	Gustav	En kontrollfråga; är du medveten om att Kent Beck vill att man ska använda allting i metoden, för att man ska kunna säga att man använder XP?
41	Lundström	Ja. Det är därför jag säger att jag inte använder XP.
42	Gustav	Ok, jag misstänkte det, jag ville bara få det bekräftat.
43	Lundström	Och, också, det är väldigt många som säger att: ”jaja, men du använder inte allting, då kommer det aldrig att funka, du måste använda tolv av tolv och du måste använda de slaviskt på det här sättet, annars kommer det aldrig att funka”. Det håller jag inte heller med om.
44	Gustav	Vad är det som gör att du inte håller med om det?
45	Lundström	Om vi tar påståendet med att man måste använda tolv av tolv och använda det slaviskt, så tycker jag faktiskt att det funkar utan att använda tolv av tolv.
46	Björn	Du har sett det i praktiken?
47	Lundström	Ja, det kan man säga. Dels så tycker jag inte att man måste följa det slaviskt, dels så tycker jag inte att man alltid måste följa exakt tolv av tolv. Jag skulle kunna tänka mig att klara mig utan parprogrammering i ett projekt, till exempel. Om det skulle vara så att jag skulle make-a-statement att jag får inte använda parprogrammering, men det känns ju lite dumt att göra det bara för att make-a-statement.
48	Gustav	Ok. Har du några erfarenheter av att försöka implementera allting, som man ska by-the-book?
49	Lundström	Nej, det har jag inte.
50	Gustav	Har du hört om någon som har försökt någonting i den stilen?
51	Lundström	Faktiskt inte heller. Ingen bekant som jag har träffat på säger att de kör allt by-the-book.
52	Gustav	Är det en uppfattning du har, att man anpassar metoden XP, rent allmänt, den erfarenhet du har fått?
53	Lundström	Jag tror faktiskt att det är så här att man utgår från andra vinkeln, snarare än att man tar in en schysst källkodshantering, man tar in en kontinuerlig integration, man tar in customer- on-site och kanske lite testande. Och sen upptäcker man att oh shit, vi använder allt som finns i XP, att istället för att säga att vi skulle köra XP, så vi måste göra det här, så tar man liksom bottom-up eller hur man ska kalla det.
54	Gustav	Så Kent Beck har samlat ett antal praktiker som han tycker fungerar bra och ska komplettera varandra, men i praktiken så har han bara hittat bra saker som andra också hittar och sen så råkar det vara lite som han har byggt det. Är det så du menar?
55	Lundström	Ja, det är väl inte så att man gör saker helt blint heller, utan man ser att den där personen gör det, och dom gör de, och dom gör de, så då borde kanske jag också titta på det.
56	Gustav	Okej. Om vi går in lite på krav. Det är egentligen vår huvudfokus i arbetet, det gäller de krav som XP ställer som metod, och man kommer ju lite in på det här i och med att man anpassar metoden, så är det på nåt sätt som att den ställer krav, kanske de inte passar eller på nåt sätt är orimliga. Så om vi då utgår från om du skulle på nåt sätt ska gå in i varför du inte följer alla tolv praktikerna. Vad skulle det krävas, kravmässigt av de individuella utvecklarna för att göra det? Kräver det för mycket, är det därför de inte gör det?
57	Lundström	Jag tror att ska det vara en duktig TDD-människa så ska det vara en jävligt duktigt mjukvarudesigner. Alltså, du ska ha bra koll på vad vad fanken du gör, man ska kunna abstrahera bort sig själv och se tre, fyra steg i förväg, vad

		det är som kommer kunna hända för annars är det väldigt lätt att det blir väldigt blint och liksom, även en blind höna finner ett korn till slut. Så ja, det är nog sant att testdriven utveckling framför allt ställer höga krav på kompetensnivån.
58	Gustav	Han har en hel del roller som han ska fylla, bland annat testare, programmerare, designer och så vidare. Det är ju att ha en bred bild av vad man ska klara av.
59	Lundström	Mm, precis. Just det här testandet har jag faktiskt varit med om och sett utvecklare, en person som skulle ta över mitt projekt, som har varit tillräckligt bra utvecklare som kan Java, .NET, och verkligen inte har hajat vad jag gjorde och varför jag gjorde det. Ska man testa så ställer det krav på ens design eftersom man måste göra saker de-coupled, vilket för visso leder till en bra design, men man måste inte göra så för att ha ett program som bara funkar. Och den här personen kunde inte abstrahera de här boxarna så mycket så att man förstod hur det här fungerade ihop, så man behöver nog vara en duktig utvecklare, dels för att kunna förstå när man ska ta över, och dels för att kunna implementera det också.
60	Gustav	Okej. Om vi går vidare till organisationen. XP ställer en hel del krav, till exempel på att ta in en customer-on-site kan vara både dyrt och svårt att övertyga, samt någon management position och så vidare. Finns det liknande situationer där, att vissa av de här praktikerna ställer höga krav på organisationen?
61	Lundström	Oh ja. Det är likadant med Scrum, som vi här på SoftHouse profilerar oss väldigt mycket på, att man ska ha en singel-point-of-contact, där den faktiskt kan ta alla beslut, i princip. Det är faktiskt en organisation som jag har träffat på som har det så, då var det en liten utvecklingsorganisation, med några få utvecklare och en produktchef som tog alla beslut. Alla andra stora organisationer är organiserade på ett helt annat sätt, där det finns tio olika personer som rent teoretiskt sätt är stakeholders i det här projektet, och alla kommer in som kommer ut från alla håll. I och för sig, nu när jag tänker på det, var jag med i ett projekt på ett jättestort företag, det var ett rätt så litet projekt, för en liten del av företaget, där det faktiskt fanns fler stakeholders, där det faktiskt fanns en sammanförande person som kunde ta de flesta besluten, men han var inte on-site jämt, för han jobbade i Danmark och kunde inte vara i Skåne varje dag. Alla de här andra är rätt så små, alltså köra testdriven utveckling, sure, det är vi i projektet som bestämmer, vi kan sätta upp en byggserver som håller vår källkod, det är inga problem, men ska man ha den personen som ställer krav, det är ofta någon från marknadsavdelningen om det är någonting som ska gå ut till end-users, det är där det stora kruxet finns rent organisationsmässigt, för de flesta företag är inte organiserade så. Det tror jag faktiskt är en huvudsak som jag tycker att ni borde trycka på, som skriver en uppsats, ni borde försöka hitta en organisation som funkar, som har en customer-on-site, eller produktägare som Scrum kallar det, där ni kan visa… jag är helt övertygad om att ett sånt företag som är organiserade kring produkter, eller projekt eller vad vi nu ska kalla det, med en person som är kravställare i mitten som kan ta alla beslut, det går så jävla mycket snabbare att höra med den personen än att man skickar iväg ett mail till Kalle där borta, och sen ska Ulla vara med och ha någonting att säga, så tar det en vecka innan man kan få ett svar.
62	Gustav	Så det är svårt att hitta en organisation där man kan ha den här allsmäktiga bestämmaren?
63	Lundström	Mmm.
64	Gustav	Om vi tittar på lite andra delar av de här organisationskraven, finns det delar i XP som stöter sig lite med vilken syn man har på hur ett projekt ska genomföras? Då tänker vi på att XP är ganska självständigt för

		programmerarna att programmera i ett XP-projekt, ganska mycket frihet. Är det några sådana delar som kan vara lite obekväma för ledningen?
65	Lundström	Ja, alltså rent generellt så brukar det ju vara så att det finns en del tekniska stakeholders i ett projekt som har, vad ska man säga, som har bestämt att vi jobbar med de här teknikerna, vi jobbar med java eller .NET, och det finns inget annat ni får jobba med, det är en sådan klassisk grej… XP säger inte vilket programspråk man ska jobba efter men om man ska empower-the-team så finns det svårigheter om teamet tycker att man ska använda java istället för .NET, eller tvärtom.
66	Gustav	De säger, så här gör vi? Då ska ni också göra så.
67	Lundström	Ja, precis. Det är inte alltid av ondo heller, det finns fördelar, framför allt maintance-mässigt, när ett projekt ska lämnas över till en ny utvecklingsorganisation så är det nog bra om det mesta är skrivet i samma språk.
68	Gustav	Vi är egentligen inte ute efter någon värdering här, utan vi är mer intresserade av krav. Om de sedan är bra eller dålig är upp till den som läser. Absolut, det är intressant, det finns olika spektrum av de här kraven, och det gäller även den individuella utvecklaren också. Vi har forskat i det här, läst många artiklar och varit på några intervjuer, och det finns en uppfattning om att XP ställer mycket höga krav, i varje fall om man skall följa det fullt ut. Det är en ganska komplicerad metod, boken han beskriver den i är 200 sidor stor.
69	Björn	Den verkar enkel att applicera men är inte det i praktiken.
70	Gustav	Har vi förstått i alla fall. Det är väldigt intressant. Vi trodde också att det här var en metod som var klart mer populär än den var. Vi har nu insett att vi nog har frågat lite fel personer, vi har frågat mer representanter från företaget eller så, och vi har fått en hint om att beslut om att använda XP tas ganska lågt ner, kanske utvecklarna själva eller projektledaren säger: vi kör de här teknikerna, jag kör testdriven först.
71	Lundström	Jag är ute på ett uppdrag just nu, där de trycker rätt så hårt på testdriven utveckling, eller rättare sagt automatiska tester, mycket unit-tester och sånt. Jag hörde en siffra, att du ska ha en code-coverage på 80 procent, för att de ska vilja ta emot din mjukvara när du väl slutar i projektet. Den kunden är faktiskt rätt så långt framme i den tekniska delen, av det som är XP, men så har de inte organiserat sig runt det här, feature-drivet, eller projekt-drivet på samma sätt, men den tekniska organisationen är rätt mogen, det finns en standard CI-miljö som man kan jacka in sina bygg-script i och sånt. Man är rätt så mogen faktiskt.
72	Gustav	Jag tänkte innan, du sa det om Scrum, ni använder mycket Scrum har vi förstått när vi pratade med Christian. XP har ju The Planning-game och Scrum har ju sin liknande variant, nu kan jag inte Scrum jättebra, men de har ju vad jag förstått en liknande variant av det här, där man har product-backlog där man ska ta in olika funktioner. Blir det så, att man använder Scrum, och sen XP lite som ett komplement?
73	Lundström	Mm, eller tvärt om.
74	Gustav	Jasså, tvärt om också?
75	Lundström	Det är lite som så att Scrum och XP är väldigt kompletterande. XP fokuserar väldigt mycket på ”hur ska vi göra det vi jobbar med?”. Alltså, du ska göra det med test-first, du ska skriva tester, sen blir det TDD till slut. Du ska göra så här, du ska ha en CI-miljö. Scrum skiter i det, de är liksom uppe på moln-nivå. Man använder Scrum för att plocka in vilka features vi ska jobba med i en iteration och så använder man XP för hur man jobbar inuti den iterationen, hur gör vi vår utveckling? Scrum är väldigt relaxed, han Southerland skiter i det princip. Southerland och Schwaber tog det bara på projektnivå, och skiter i det på utvecklingsnivån. När man använder Scrum, utan att använda XP-practices så brukar man göra det i några månader, sen går man tillbaks till det

		man hade innan, eftersom det inte går så bra, märker man. Det gör nämligen väldigt ont i Scrum om man ska släppa en feature inkrement vid varje iteration som är potentiellt installerbar och man måste manuellt testa hela det inkrementet. Man lägger fantastiskt mycket tid på det om man ska testa till varje sprint, det är därför man använder automatiska tester för att slippa göra de manuella grejerna vid varje sprint.
76	Björn	Det verkar nästan som att man tvingas att blanda metoder, att det är väldigt svårt att använda Scrum etthundra procent och väldigt svårt att använda XP etthundra procent. Antingen en mix, eller någonting i kombination med nått annat.
77	Lundström	Ja, precis. De jobbar på olika nivåer, på nått sätt.
78	Gustav	Är det här en svaghet i metoderna?
79	Lundström	Det tycker jag inte, snarare en styrka. Givet att du vet vad du sysslar med, i någon mån i alla fall så kan du faktiskt ta Scrum och XP och du kan liksom tweaka till dem så som du känner att du vill jobba. En del i Scrum är den här kontinuerliga reflektionen, jag vet inte om XP också specificerar det, men att man i slutet av varje iteration ska man titta bak i ett kort retrospektiv, vad var det vi gjorde bra, vad var det vi gjorde dåligt, och i och med det så har man faktiskt verktyg för att tweaka till sin process, dels på Scrum-nivå, dels på nivån vad är det vi gör, bör vi börja med test-first, bör vi sluta med vår parprogrammering för det slänger bara bort tid.
80	Gustav	Om vi tar lite mer generell karaktär på frågorna: vad är positivt med XP?
81	Lundström	Jag tycker att man får mer mjukvara på kortare tid med högre kvalitet av färre personer, än klassiska …
82	Björn	Vattenfall?
83	Lundström	Ja, alltså, vattenfall och XP är ju inte motstående heller. XP säger att man ska köra iterationer, vilket i och för sig är motstående mot vattenfall, men det finns ju andra hemkok på processer också.
84	Gustav	Ok, men vad tycker du är negativt med XP?
85	Lundström	Det ställer krav på utvecklarna framför allt, och det kan ge gnissel i organisationen. Dels på produktägarfrågan, det kan vara svårt för XP att fungera när man har en organisation som inte är orienterad kring det. Å andra sidan så finns det förhoppningsvis sätt att ta sig runt det så man funkar nästan rätt så bra i alla fall. Som utvecklare, man kan inte sätta en helt grön person på att göra tesdriven development. Ett test kan man visa jätteenkelt, så här funkar det i JUnit, och det är väldigt enkelt, det är bara till att skriva test, men den personen kommer inte kunna göra något vettigt på ett år om man inte stödjer den personen.
86	Gustav	Så man behöver vara en ganska komplett utvecklare för att kunna följa metoden som den är tänkt att följas? Är det så man skulle kunna tolka det?
87	Lundström	Ah, det kan man tänka sig. Jag gillar det som brukar kallas domain-driven design, alltså organisera sig kring sin domänmodell, och sättet som man pratar till varandra, både i teamet och med varandra som utvecklare, och sen med den som är kundkontakten. Man måste vara en väldigt duktig kommunikatör, för att vara en duktig utvecklare. Det är inte så att du ska ha en person som hanterar alla dina krav, som sen ska lämna över en stor mängd krav till dig, utan du som utvecklare måste faktiskt leta reda på vad är det vi behöver kunna göra, så du har helt rätt, man måste vara väldigt komplett som utvecklare.
88	Gustav	Då har vi en fråga som lyder såhär: Vad är din åsikt om XP, tycker du om det, tycker du inte om det?
89	Lundström	Jag gillar det.
90	Gustav	Ok, du gillar det. Vad är det som gör att du gillar det?
91	Lundström	Man blir produktiv om man använder de practices som finns i XP.
92	Gustav	Ok. Det är en viktig kontrollfråga för oss, när vi ska utvärdera det här bias och

		så.
93	Björn	Vi kan ta en avslutande reflektion om framtiden: Vad tror du om XP i framtiden, eller agila metoder överlag? Hur tror du det kommer se ut om en två-tre år någonting?
94	Lundström	Jag vet inte riktigt hur mycket ni hänger med vad som händer runt om i företagen, men just nu, as we speak, så är det en väldigt stor våg av agile, vad man nu kan köra; XP, Scrum eller något annat.
95	Björn	Det har vi förstått ja.
96	Lundström	Precis, så jag tror att om en två-tre år så kommer det att vara en hel del som har bränt sig, för att de inte hade den kunskapen, de var inte mogna för att gå dit. Nu är det stora företag med tusentals anställda som går över och säger att nu ska vi vara agile, det finns en rätt så stor risk att det kommer implodera. Å andra sidan är det väldigt spännande för vi som lyckas, vi kommer kunna göra jävligt mycket på jävligt kort tid, med jävligt hög kvalitet. Så det är positivt, helt klart.
97	Gustav	Det finns en framtid för det?
98	Lundström	Definitivt.
99	Gustav	Det är intressant, vi har fått precis samma åsikter från två andra, som du säger, att man kommer försöka, men misslyckas precis som du säger. Har du några avslutande reflektioner, något du vill få med, som vi har missat?
100	Lundström	Nej, jag vet inte, en av grejerna, fast den har jag redan fått med, det är testdriven utveckling. Det är inte någon silverbullit, det kommer inte ge någon bra arkitektur, inte någon god design, utan du behöver göra lite design up-front, lite arkitektur up-front, för då vet du vad det är du ska göra. Däremot så behöver du kanske inte lägga skitmycket tid på det, absolut inte mer än några dagar. Det tillhör början av ett projekt, man snackar krav, man får en arkitektur som kanske kommer funka. Det är nämligen det, om man snackar TDD, så är det väldigt många som säger att bara man gör TDD så kommer allting att lösa sig, och då är man ute på väldigt djupt vatten. Man kan misslyckas rätt så hårt med TDD också.

B6) The agile manifesto

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Referenser

Abrahamsson, P. (2003). Extreme programming: first results from a controlled case study. *Euromicro Conference, 2003. Proceedings. 29th* , pp. 259-266.

Abrahamsson, P., & Koskela, J. (2004). Extreme programming: a survey of empirical data from a controlled case study. *Empirical Software Engineering, 2004. ISESE '04. Proceedings. 2004 International Symposium on* , pp. 73-82.

Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003). New directions on agile methods: a comparative analysis. *Software Engineering, 2003. Proceedings. 25th International Conference* , pp. 244-254.

Agile Alliance: What Is Agile Software Development? (2006). Retrieved 11 25, 2008, from Agile Alliance: http://www.agilealliance.org/show/2

Agile Sweden - nätverket för flexibla systemutvecklingsmetoder. (n.d.). Retrieved November 25, 2008, from Agile Sweden: http://www.agilesweden.org/

Atkinson, R. (1999). Project management: Cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. *International Journal of Project Management* , pp. 337-342.

Avison, D., & Fitzgerald, G. (2006). *Information systems development.* Berkshire: McGraw Hill Education.

Bajec, M., Vavpotic, D., & Krisper, M. (2007). Practice-driven approach for creating project-specific software development methods. *Information and Software Technology* , 49, p. 345.

Beck, K. (1999). Embracing change with extreme programming. *Computer , 32 Nummer 10*, pp. 70-77.

Beck, K. (2000). *Extreme Programming Explained.* Reading, MA: Addison-Wesley.

Beck, K., & Andres, C. (2005). *Extreme Programming explained - Embrace change* (2:a upplagan ed.). Boston: Addison-Wesley.

Bendix, L., Magnusson, B., & Hedin, G. (2005). Teaching extreme programming to large groups of students. *The Journal of Systems and Software* , 74, pp. 133-146.

Bevan, N., Kirakowskib, J., & Maissela, J. (1991). What is Usability? *Proceedings of the 4th International Conference on HCI.* Stuttgart.

Boivie, I., Gulliksen, J., & Göransson, B. (2006). The lonesome cowboy: A study of the usability designer role in systems development. *Interacting with Computers* , 18, pp. 601-634.

Chow, T., & Cao, D.-B. (2008). A survey study of critical success factors in agile software projects. *The Journal of Systems and Software* , 81, p. 961.

Coleman, G. (2004). eXtreme Programming (XP) as a 'minimum' software process: a grounded theory. *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual Internationa* , pp. 30-31 vol.2.

Dick, S. J. (2000, Februari 24). *X-15 - Hypersonic Research at the Edge of Space.* Retrieved 11 25, 2008, from NASA.gov: http://history.nasa.gov/x15/cover.html

Extreme Programming Explained: Embrace Change: Kent Beck: Amazon.co.uk: Books. (n.d.). Retrieved 11 17, 2008, from Amazon.co.uk.

Fitzgerald, B., Hartnett, G., & Conboy, K. (2006). Customising agile methods to software practices at Intel Shannon. *European Journal of Information Systems* , 15, pp. 200-213.

Fitzgerald, B., Russo, N. L., & Stolterman, E. (2002). *Information Systems Development Methods in Action.* New York: McGraw-Hill Education.

Fruhling, A., McDonald, P., & Dunbar, C. (2008). A Case Study: Introducing eXtreme Programming in a US Government System Development Project. *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)* , pp. 464-464.

Hartson, R. H. (1998). Human-computer interaction: Interdisciplinary roots and trends. *The Journal of Systems and Software* , 43, pp. 103-118.

Hedin, G., Bendix, L., & Magnusson, B. (2003). Introducing software engineering by means of extreme programming. *Software Engineering, 2003. Proceedings. 25th International Conference on* , pp. 586-593.

Highsmith, J. (2001). *History: The Agile Manifesto.* Retrieved 11 25, 2008, from Manifesto for Agile Software Development: http://www.agilemanifesto.org/history.html

Highsmith, J., & Cockburn, A. (2001). Agile software development: the business of innovation. *Computer* , 34, pp. 120-127.

Hussain, Z., Lechner, M., Milchrahm, H., Shahzad, S., Slany, W., Umgeher, M., et al. (2008). Optimizing extreme programming. *2008 International Conference on Computer and Communication Engineering* , pp. 1052-1056.

Jackson, A., Shiu Lun, T., Gray, A., Driver, C., & Clarke, S. (2004). Behind the rules: XP experiences. *Agile Development Conference, 2004* , pp. 87-94.

Jacobsen, D. I. (2002). *Vad, hur och varför? - Om metodval i företagsekonomi och andra samhällsvetenskapliga ämnen.* Studentlitteratur.

Jones, C. (2000). *Software Assessments, Benchmarks, and Best Practices.* Boston, MA: Addison Wesley.

Karlström, D. (2002). *Introducing Extreme Programming - An Experience Report.* Lund: Dept. Communication Systems, Lund University.

Keenan, F. (2004). Agile process tailoring and problem analYsis (APTLY). *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on* , pp. 45-47.

Kent Beck. (n.d.). Retrieved November 18, 2008, from Three Rivers Institute: http://www.threeriversinstitute.org/Kent%20Beck.htm

Kvale, S. (1997). *Den kvalitativa forskningsintervjun.* Studentlitteratur .

Larman, C., & Basili, V. R. (2003). Iterative and incremental development: a brief history. *Computer , 36*, pp. 47-56.

Lawrence, R. (2007). XP and Junior Developers: 7 Mistakes (and how to avoid them). *AGILE 2007* , pp. 234-239.

Layman, L., Williams, L., & Cunningham, L. (2004). Exploring extreme programming in context: an industrial case study. *Agile Development Conference, 2004* , pp. 32-41.

Leth, G., & Thurén, T. (2000). *Källkritik för Internet.* Stockholm: Styrelsen för psykologiskt försvar.

Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., et al. (2004). Agile Software Development in Large Organizations. *Computer* , 37, pp. 26-34.

Luck, G. (2004). Subclassing XP: breaking its rules the right way. *Agile Development Conference, 2004* , pp. 114-119.

Manifesto for Agile Software Development. (n.d.). Retrieved June 15, 2009, from Manifesto for Agile Software Development: http://www.agilemanifesto.org/

Mirakhorli, M., Rad, A. K., Aliee, F. S., & Pazoki, M. (2008). RDP Technique: Take a Different Look at XP for Adoption. *19th Australian Conference on Software Engineering (aswec 2008)* , pp. 656-662.

Muller, M. M., & Tichy, W. F. (2001). Case study: extreme programming in a university environment. *Software Engineering, 2001. ICSE 2001. Proceedings of the 23rd International Conference on* , pp. 537-544.

Murru, O., Deias, R., & Mugheddu, G. (2003). Assessing XP at a european internet company. *IEEE Software* , pp. 37-43.

Plamondon, S. (2003, Juni 17). *Working smarter, not harder: An interview with Kent Beck*. Retrieved November 18, 2008, from IBM.com: http://www.ibm.com/developerworks/java/library/j-beck/

Rising, L., & Janoff, N. S. (2000). The Scrum software development process for small teams. *IEEE Software , 17*, pp. 26-32.

Rumpe, B., & Schröder, A. (2002). Quantitative Survey on Extreme Programming Projects. *Third International Conference on Extreme Programming and Flexible Processes in Software Engineering, XP2002* .

Schuh, P. (2001). Recovery, redemption, and extreme programming. *IEEE Software* , 18, pp. 34-41.

Seffah, A., Donyaee, M., Kline, R. B., & Padda, H. K. (2006). Usability measurement and metrics: A consolidated model. *Software Quality Journal* , 14, pp. 159-178.

Sfetsos, P., Angelis, L., & Stamelos, I. (2006). Investigating the extreme programming system--An empirical study. *Empirical Software Engineering* , 11, pp. 269-301.

Slaughter, S. A., Harter, D. E., & Krishnan, M. S. (1998). Evaluating the cost of software quality. *Association for Computing Machinery. Communications of the ACM* , 41, pp. 67-73.

Thong, J. Y., Hong, W., & Tam, K. Y. (2004). What leads to user acceptans of digital libraries. *Association for Computing Machinery. Communications of the ACM* , 47, pp. 78-83.

Thurén, T. (2005). *Källkritik.* Stockholm: Liber.

van Welie, M., van der Veer, G. C., & Eliëns, A. (2001). Breaking down Usability. Amsterdam: Faculty of Computer Science, Vrije Universiteit Amsterdam.

Williams, L., Layman, L., & Krebs, W. (2004). *Extreme Programming Evaluation Framework for Object-Oriented Languages.* Retrieved from North Carolina State University: ftp://ftp.ncsu.edu/pub/unity/lockers/ftp/csc_anon/tech/2004/TR-2004-18.pdf